VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Mechanizmy řízení robotického auta NXP (FREESCALE)

Driving mechanisms of robotic car NXP (FREESCALE)

2019 Richard Zvonek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Richard Zvonek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Mechanizmy řízení robotického auta NXP (FREESCALE)
Driving Mechanizms of Robotic Car NXP (FREESCALE)

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit software pro ovládání robotického auta FREESCALE s kitem FRDMZKL25Z. Software umožní účast na soutěži NXP Cup a připojení nadřazeného řídícího modulu Raspberry Pi. Řešení se bude zabývat těmito oblastmi:
1. Zpracování dat z řádkové kamery.
2. Detekce černých okrajových čar.
3. Načítání a filtrace dat ze senzorů (AD převodník, mikroakcelerometr).
4. Propočet polohy auta v prostoru.
5. Rozšíření HW auta o další senzory (např. otáčkoměr).

Práce bude obsahovat:
1. Přehled používaných metod a algoritmů.
2. Implementaci výše popsané funkcionality.
3. Experimenty a vyhodnocení výsledků.
4. Programátorskou dokumentaci řešení s využitím diagramů jazyka UML.

Seznam doporučené odborné literatury:

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025
[2] DARWIN, Ian F. Java cookbook. 2nd ed. Sebastopol, CA: O'Reilly, c2004, xxiv, 829 p. ISBN 05-960-0701-9. Dostupné z: http://it-ebooks.info/book/2249/

Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 3. května 2019 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 3. května 2019 .....

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Davidovi Ježkovi, Ph.D. a Ing. Petrovi Olivkovi, Ph.D. za pomoc při vypracování práce, jejich věcné připomínky, zapůjčení hardwaru a dodání softwarového základu, o který se tato práce opírá.

Abstrakt

Tato bakalářská práce se zabývá možnými metodami řízení robotického modelu auta za pomocí dat získaných z řádkové kamery a následného výpočtu pozice modelu auta na závodní dráze. Práce konkrétně rozebírá jednotlivé aspekty potřebné pro vytvoření funkčního, konkurenceschopného softwaru pro autonomní model auta pro použití v soutěži NXP Cup. Zároveň je v práci popsán způsob práce s přidanými senzory, nebo možnost bezdrátového přenosu dat. Oproti standardnímu modelu auta pro NXP Cup je přidán jednodeskový počítač Raspberry Pi. Software pro model auta je vytvořen v jazyce C++, vytvořený software je poté možné zkompilovat jak pro Raspberry Pi, tak pro vybrané mikropočítače firmy NXP.

Klíčová slova: Autonomní vozidlo, NXP Cup, FRDM-K66F, Raspberry Pi, IMU, IR senzor

Abstract

This bachelor thesis is concerned with possible methods of controlling a robotic car model with the aid of data gained from a line scan camera and subsequent calculation of the car model position on race track. The thesis analyses individual aspects which are necessary to make a functional and competitive software for autonomous car model that can be used in NXP Cup competition. There is a description of work with added sensors in the thesis as well, or a possibility of wireless transmission of data. In comparison with a standard car model for NXP Cup there is a single-board computer Raspberry Pi added. The software of the car model is made in a C++ language. This software can then be compiled for both Raspberry Pi and selected NXP microcomputers.

Key Words: Autonomous vehicle, NXP Cup, FRDM-K66F, Raspberry Pi, IMU, IR sensor

Obsah

Seznam použitých zkratek a symbolů

ADC	–	Analog to Digital Converter
ARM	–	Advanced RISC Machine
CSV	–	Comma-separated Values
DHCP	–	Dynamic Host Configuration Protocol
DMP	–	Digital Motion Processor
EMEA	–	Europe, Middle East, Africa
GPIO	–	General-Purpose Input/Output
GPS	–	Global Positioning System
I2C	–	Inter-Integrated Circuit
IMU	–	Inertial Measurement Unit
LED	–	Light Emitting Diode
MCU	–	Microcontroller Unit
MPU	–	Microprocessor Unit
PWM	–	Pulse Width Modulation
RAM	–	Random Access Memory
RF	–	Radio Frequency
RGB	–	Red Green Blue
RISC	–	Reduced Instruction Set Computer
SD	–	Secure Digital
SSH	–	Secure Shell
SWD	–	Serial Wire Debug
TFC	–	The Freescale Cup
UART	–	Universal Asynchronous Receiver-Transmitter
UDP	–	User Datagram Protocol
USB	–	Universal Serial Bus
VPN	–	Virtual Private Network
Wi-Fi	–	Wireless Fidelity
XML	–	eXtensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Řízení automobilu byla donedávna disciplína, která nebyla automatizována a pro řízení byla tradičně vždy potřeba lidská síla. V současně době se postupně řízení automatizuje a upouští se od plného lidského ovládání automobilů. Ať už jde o asistenční prvky typu adaptivních tempomatů, nebo systémů udržování v jízdním pruhu, nebo o plně autonomní řízení, se kterým v současnosti experimentuje velká část světových automobilek. Přestože v současné době existují automobily, které se tváří jako autonomní, je zatím potřeba, aby byl vždy přítomen řidič, který musí být kdykoliv připraven převzít řízení.

Tato bakalářská práce popisuje jednoduchý systém pro robotický model auta, který se v prostoru orientuje pomocí řádkové kamery. Systém je schopný bezdrátově přenášet data do počítače, ve kterém jsou vizualizovány aktuální informace o řízení a obraz z kamery. Zároveň je možné k modelu auta připojit jednodeskový počítač Raspberry Pi pro vyšší výpočetní výkon, nebo pro zjednodušené nastavení různých řídících parametrů.

Práce se nejprve zabývá popisem soutěže NXP Cup a jejími pravidly, dále je popsán hardware robotických modelů aut, které byly při vypracování použity a na kterých byl vznikající systém testován. Následně je věnována krátká kapitola podpůrné desktopové aplikaci pro zobrazení obrazu z kamery a informací o aktuálních řídících parametrech. V následující softwarové části je popsána komunikace se zmíněnou aplikací, abstrakce systému pro zajištění přenositelnosti a hlavní části věnované zpracování obrazu a řízení modelu auta pomocí dat ze zpracovaného obrazu. Následuje popis softwarové stránky použitých senzorů, experimentální část a postřehy z účasti v soutěži NXP Cup.

Veškerý software vzniklý při vypracování této práce je psán v jazyce C++, při tvorbě softwaru bylo použito vývojové prostředí Visual Studio pro vývoj desktopové aplikace, CLion pro vývoj společných částí aplikace a pro vývoj Linuxové verze pro Raspberry Pi; a MCUXpresso IDE pro vývoj embedded části pro mikropočítač FRDM-K66F. Pro verzování byl použit systém Git.

2 NXP Cup

NXP Cup je globální soutěž v řízení autonomních vozidel. Česká republika spadá do kategorie EMEA. Regionální kola se konaly v Ostravě, Paříži, Mnichově, Bejrútu, Bukurešti a Casablance. Finální kolo se konalo v Norimberku. Pro úspěšnou účast je potřeba skloubit znalosti programování pro embedded zařízení, návrh a realizaci regulátorů, hardwaru a zpracování obrazu. Většina použitého hardwaru se řadí do kategorie tzv. low cost zařízení - tzn. hardware a periferie s nízkou cenou.

2.1 Nová pravidla pro rok 2019

Během vypracování práce se změnily pravidla pro rok 2018. Software nebylo potřeba z větší části předělávat, ale bylo potřeba adaptovat nastavení pro nová pravidla a implementovat řízení pro dodatečné disciplíny. Úpravy pravidel pro rok 2019 oproti předchozím ročníkům.[10]

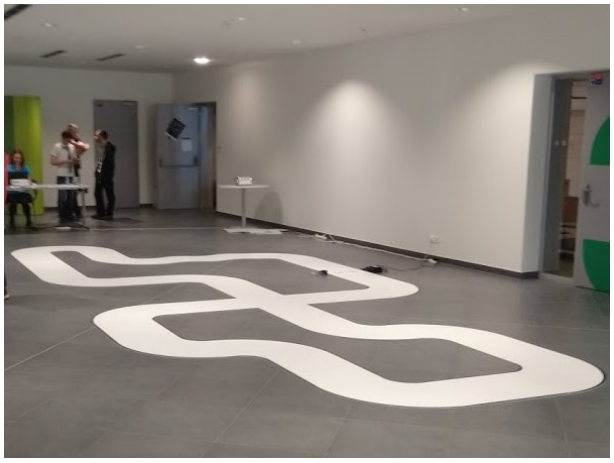
- Tým je tvořen dvěma nebo třemi studenty. Členové týmu mohou studovat jakýkoliv obor. Tým může být tvořen studenty libovolných ročníků.
- Není omezen počet dodatečných senzorů. Je možné přidat libovolné množství senzorů, senzory firmy NXP musí být použity, pokud jsou společností NXP dodávány.
- Je povoleno využít libovolné MCU nebo MPU, nebo kombinaci obou. Všechny vývojové desky musí být vytvořeny firmou NXP, nebo musí obsahovat MCU/MPU firmy NXP.
- Šířka dráhy je zmenšena z 60cm na 55cm. Zároveň je oficiální dráha tvořena novým materiálem.
- Byly přidány nové dodatečné disciplíny.
- Není potřeba dodávat technickou zprávu ohledně změn vůči oficiálnímu kitu. Je ale potřeba dodat seznam změn.
- Auta s podvozky typu Alamak i Model C soutěží ve stejné kategorii.

2.2 Disciplíny

Pro úspěšné umístění se v NXP Cup je ideálně potřeba zvládnout všechny disciplíny. Všechny disciplíny jsou hodnoceny bodově, celkový součet bodů ovlivňuje pořadí soutěžních týmů. Hlavní závod je pro všechny účastníky povinný a je možné zde získat nejvíce bodů. Ostatní úkoly jsou nepovinné, ale ziskem bodů ovlivní finální pořadí. Hlavní závod i dodatečné disciplíny probíhají na stejném typu dráhy.

2.2.1 Závod

Hlavní disciplínou v NXP Cup je samotný závod. Závodí se na dráze, která může obsahovat všechny standardní segmenty dráhy. Finální sestavení dráhy není před závodem známo a není možné na finálně sestavné dráze trénovat. Při závodě se hodnotí čas, je třeba zajet kolo co nejrychleji. Při závodě nesmí auto vyjet z dráhy více než dvěma koly. Na závod má každý tým tři pokusy a hodnocen je první úspěšný pokus. Po úspěšném pokusu už není možné vylepšit čas dalším pokusem. Mezi jednotlivými pokusy má každý tým čas 60 vteřin na úpravu parametrů pomocí např. přepínačů nebo potenciometrů na desce auta. Mezi jednotlivými pokusy není možné přeprogramovat software.



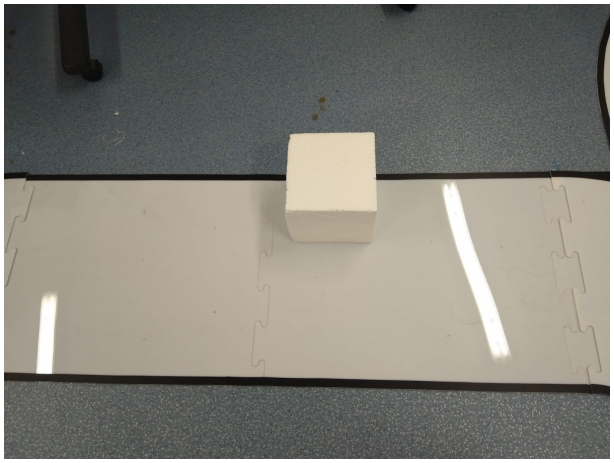
Obrázek 1: Sestavená dráha pro hlavní závod kvalifikačního kola

2.2.2 Vyhnutí se překážce

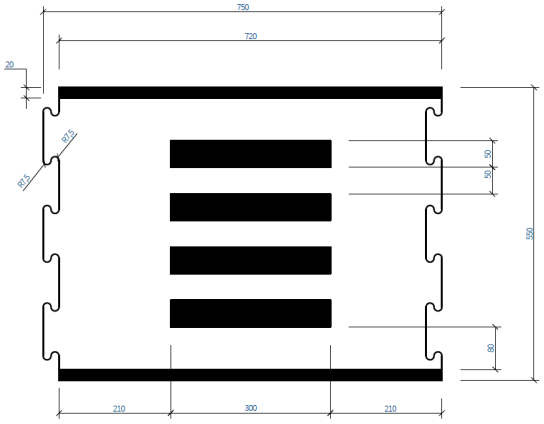
Disciplína vyhnutí se překážce se odehrává na malé oválné trati, na které je položena bílá kostka o velikosti 20x20x20cm (viz Obrázek 2). Auto nesmí ani částí kola vyjet z dráhy, jinak je pokus neplatný. Auto musí nejprve objet jeden okruh na dráze a poté je na dráhu položena překážka. Překážka může být umístěna kdekoliv na dráze. Pokud se auto překážce vyhne, bere se pokus jako úspěšný. Na splnění disciplíny má každý tým pouze jeden pokus.

2.2.3 Omezení rychlosti

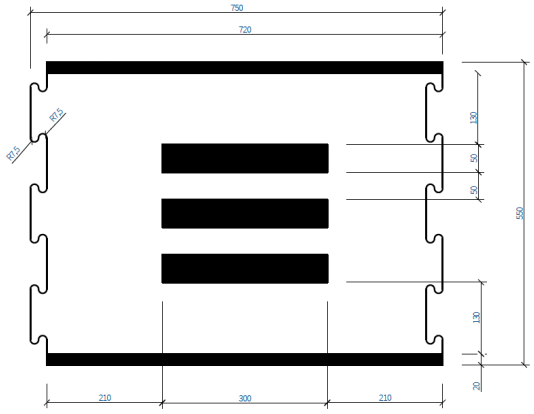
Pro tuto disciplínu je potřeba rozpoznat nové typy trati - zóna zpomalení (viz. Obrázek 3) a konec zóny zpomalení (viz. Obrázek 4). Na krátké oválné trati jsou umístěny nové segmenty trati a při přejezdu zóny zpomalení je potřeba viditelně zpomalit rychlost auta (zhruba na polovinu původní rychlosti). Při přejezdu segmentu konce zpomalení je třeba zvýšit rychlost auta na původní rychlost. V této disciplíně má každý tým pouze jeden pokus, při kterém auto nesmí vyjet žádnou částí kola ven z trati. Trať musí být objeta do 90 vteřin od začátku pokusu.



Obrázek 2: Překážka na dráze



Obrázek 3: Začátek zóny zpomalení



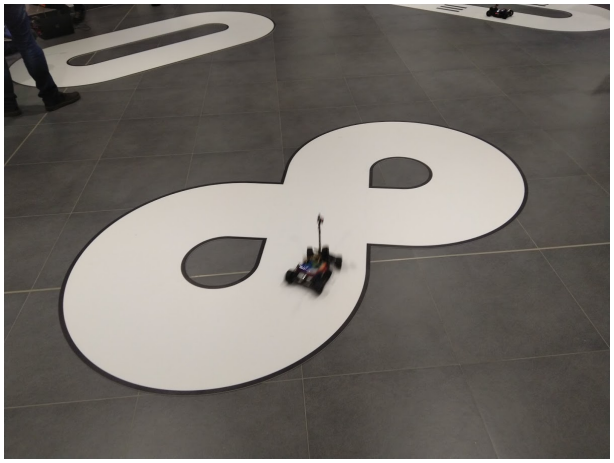
Obrázek 4: Konec zóny zpomalení

2.2.4 Počet kol v "osmičce"

V této disciplíně je důležitá přesnost a spolehlivost systému. Dráha je postavena do tvaru čísla "8"(viz Obrázek 5). Cílem disciplíny je objet co nejvíce kol v časovém limitu 90 vteřin. Narozdíl od hlavního závodu nesmí auto vyjet z dráhy ani částí kola. Pro úspěšné zvládnutí disciplíny má každý tým tři pokusy, počítá se první úspěšný pokus. Mezi jednotlivými pokusy má každý tým čas 60 vteřin na úpravu parametrů.

2.2.5 Finální hodnocení

Hlavní závod a závody v osmičce jsou hodnoceny bodově podle pořadí. V hlavním závodě je hodnocen nejlepší čas, v osmičce počet dokončených kol. V případě shodných výsledků dvou a více týmů získávají týmy stejný počet bodů. Bodování hlavního závodu a závodu v osmičce je zobrazeno v Tabulce 1. Disciplíny omezení rychlosti a vyhnutí se překážce jsou hodnoceny



Obrázek 5: Tvar osmičkové dráhy v kvalifikačním kole

shodně - 150 bodů za úspěšné dokončení, 0 bodů v případě neúspěchu. Celkem je tedy možné získat maximálně 1000 bodů.

Nejlepší čas	Hlavní závod	Nejvíce kol	Osmička
1.	500	1.	200
2.	400	2.	150
3.	350	3.	125
4.	300	4.	100
5.	250	5.	75
6.	200	6.	50
7.	150	7.	40
8.	100	8.	30
9.	50	9.	20
10.	25	10.	10

Tabulka 1: Bodové hodnocení závodu a osmičky

3 Hardwarový popis modelu auta

Pro soutěž autonomního řízení NXP Cup existují dva typy standardizovaných podvozků. Prvním, starším typem je tzv. Model C. Druhým, novějším typem je tzv. Alamak. Při vypracování práce byly používány oba typy podvozků. Pro verzi auta s Raspberry Pi byl použit podvozek typu Model C, pro závody s FRDM-K66F byl použit podvozek typu Alamak. Jednotlivé podvozky jsou dále rozebrány v kapitolách 3.1.1 a 3.2.1

3.1 Model auta s podvozkem typu Alamak

Model auta s podvozkem typu Alamak bylo použito při samotném závodě soutěže NXP Cup. Kromě samotného základu - podvozku typu Alamak- auto obsahuje řídící vývojovou desku FRDM-K66F s POLI TFC shieldem, řádkovou kameru, Wi-Fi modul a tři infračervené senzory.

3.1.1 Podvozek typu Alamak

Podvozek typu Alamak je novější typ standardního podvozku pro použití v soutěži NXP Cup. Oproti staršímu typu Model C je samotná konstrukce podvozku tvořena pouze jedním kusem materiálu, celková konstrukce narostla do výšky o jeden centimetr, má výkonnější motory a disponuje většími koly. Rozvor kol zůstal zachován, stejně jako celková šířka a délka podvozku.

Podrobné technické parametry podvozku Alamak jsou vypsány v Tabulce 2, fotografie soutěžního auta s podvozkem typu Alamak je vyobrazen na Obrázku 6.

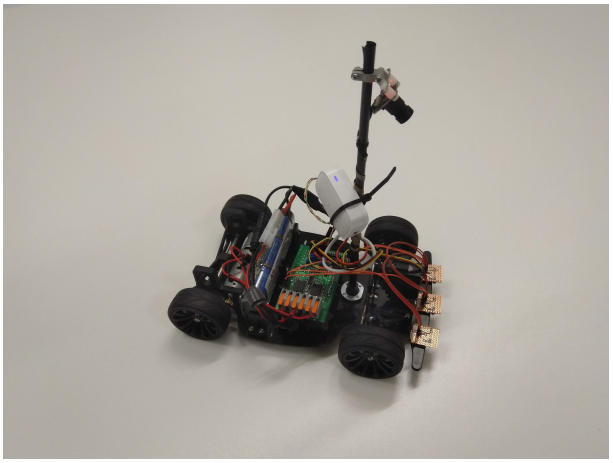
Podvozek standardně obsahuje také servomotor, který se používá pro ovládání natočení kol. Pro snímání obrazu modelu auta je pro NXP Cup použita řádková kamera. Řádková kamera snímá pouze jeden řádek obrazu. Snímání obrazu je dále rozebráno v kapitole 5.5.

Výška	80 mm
Šířka	160 mm
Délka	285 mm
Rozvor kol	160 mm
Rozchod kol	200 mm
Průměr kol	65 mm
Pohon	2x DC motor třídy 380, 7.2V
Maximální natočení kol	40°

Tabulka 2: Technické parametry podvozku typu Alamak

3.1.2 FRDM-K66F

FRDM-K66F je levná vývojová platforma vyráběná společností NXP pro MCU řady Kinetis K66, K65 a K26. Srdcem platformy je mikrokontroler ARM® Cortex®-M4, konkrétně MK66FN2M0VMD18 MCU s taktem 180 MHz, 2MB Flash paměti a 256 KB paměti RAM.



Obrázek 6: Auto s podvozkem typu Alamak

Pro konektivitu slouží dva Micro-B USB konektory, Ethernet rozhraní a 54 vyvedených GPIO pinů. GPIO piny jsou Arduino kompatibilní, je tedy možné na desku připojit libovolný shield určený původně pro platformu Arduino. Pro dodatečnou konektivitu se na desce nachází neosazená rozhraní pro RF modul `RF24L01` a pro Bluetooth modul `JY-MCU BT`.

Mezi další užitečné periferie na desce patří např. konektor pro připojení Micro SD karet, audio rozhraní, nebo SWD rozhraní pro připojení JLink ladicí rozhraní. Dále deska poskytuje dvě vestavěná tlačítka pro uživatelské vstupy a RGB LED.

Na desce také nalezneme akcelerometr kombinovaný s magnetometrem `FXOS8700CQ` a gyroskop `FXAS21002`, kterým je dále věnována kapitola 6.1.3.[5]

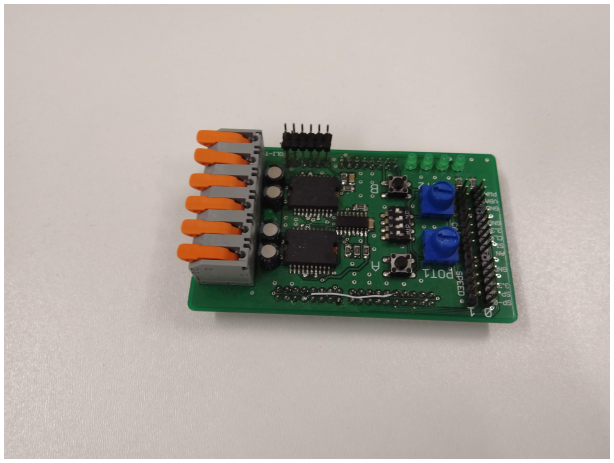
3.1.3 POLI TFC shield

POLI TFC shield je rozšiřující deska pro FRDM-K64F, resp. pro FRDM-K66F, která je odvozena od oficiálního TFC shield pro desku FRDM-KL25Z (viz kapitola 3.2.3). Jelikož jednotlivé FRDM platformy nemají plně kompatibilní piny, je potřeba mít pro každou danou platformu specializovaný TFC shield. Funkčnost jednotlivých rozšiřujících shield desek je poté na všech platformách shodná.

Pro možnost připojení infračervených senzorů bylo potřeba upravit POLI TFC shield přepájením některých pinů na průchozí.

3.1.4 Wi-Fi modul

Pro ladění nastavení parametrů a pro vizualizaci dat z kamery a senzorů je potřeba dodatečná desktopová aplikace, ve které jsou jednotlivé hodnoty zobrazeny. Pro tyto účely byla vytvořena aplikace NXP Car Interface (viz kapitola 4.1). Jelikož není praktické zobrazovat data jen pokud je auto připojeno k počítači, je potřeba odesílat data bezdrátově. Pro bezdrátový přenos dat byla vybrána technologie Wi-Fi, pomocí levného, nízkopříkonového routeru `Nexx wt3020f`, jeli-



Obrázek 7: Upravený POLI TFC shield

kož je vývojová deska FRDM-K66F vybavena vestavěným ethernet rozhraním. Wi-Fi modul je nakonfigurován jako bridge mezi výkonným routerem a samotnou vývojovou deskou. Softwarové stránce bezdrátového přenosu dat je věnována kapitola 5.1.

3.1.5 Infračervený senzor

Jedním z problémů, které je potřeba vyřešit je snímání přejetí čáry. Dle pravidel pro NXP Cup je potřeba po přejetí cílové čáry zastavit. Nezastavení po přejetí cílové čáry je trestáno penalizací jedné vteřiny [10]. Zároveň se při testování nastavení parametrů stává, že z důvodu nesprávně zvolených parametrů auto vyjede mimo trať.

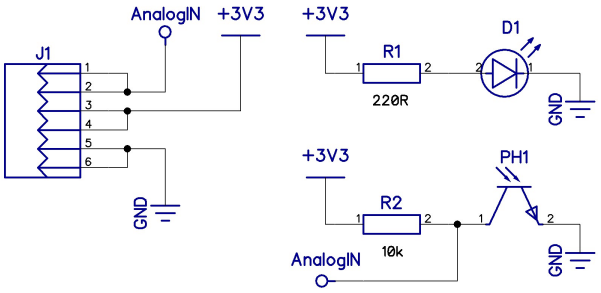
Při řešení problému detekce přejetí cílové čáry, nebo vyjetí z dráhy se ukázalo, že detekovat popsané situace pouze pomocí dat z kamery není spolehlivé. Pro zvýšení spolehlivosti a zjednodušení softwarového řešení byl navržen samostatný senzor.

Jádrem senzoru je infračervená LED, která vyzařuje infračervené záření a fototranzistor, který detekuje světlo v infračerveném spektru. Schéma zapojení a realizace senzoru lze nalézt na Obrázku 8 a 9.

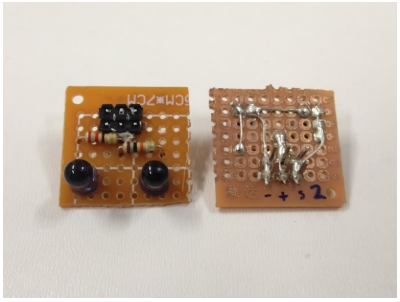
Senzor pracuje na jednoduchém principu odrazu infračervených vln od různých povrchů. Detekce přejetí čáry spočívá v tom, že na světlém povrchu je infračervené záření odráženo, zatímco na tmavém povrchu je pohlcováno. Měřením napětí na kolektoru jsme schopni zjistit, jestli se světlo z infračervené diody pohlcuje, nebo odráží – tzn. jestli je povrch pod senzorem světlý nebo tmavý.

3.2 Model auta s podvozkem typu Model C

Model auta s podvozkem typu Model C byl použit v první části vypracování této práce a dále pak při testování nastavení všech parametrů, které jsou shodné mezi jednotlivými modely (např. velikost mediánového filtru, počet uchovaných záznamů v paměti, nebo experimenty s



Obrázek 8: Schéma zapojení IR senzoru



Obrázek 9: Prototyp IR senzoru

rozpoznáním jednotlivých částí trati). Kromě základního podvozku a řádkové kamery obsahuje tento model auta vývojovou desky FRDM-KL25Z s oficiálním TFC shieldem a jednodeskový mikropočítač Raspberry Pi.

3.2.1 Podvozek typu Model C

Podvozek typu Model C je starší typ standardního podvozku, který byl používán v dřívějších ročnících soutěže a který má stále ještě oficiální podporu pro NXP Cup. Postupně je ale soutěž připravována na vyřazení podvozku typu Model C. Oproti podvozku typu Alamak je Model C výrazně méně výkonný a celkově složitější. Jeho konstrukce je tvořena více kusy a nalezneme např. i jednoduchý systém odpružení. Systém odpružení v aktuálních ročnících soutěže NXP Cup už nemá smysl, protože na dráze se již nenachází kopec. V dřívějších ročnících soutěže soutěžila auta s podvozky Alamak a Model C na oddělených tratích, ve vlastních kategoriích. Letošní ročník je přechodový, kde oba typy aut soutěží ve stejné kategorii. Technické parametry podvozku typu Model C jsou zobrazeny v Tabulce 3

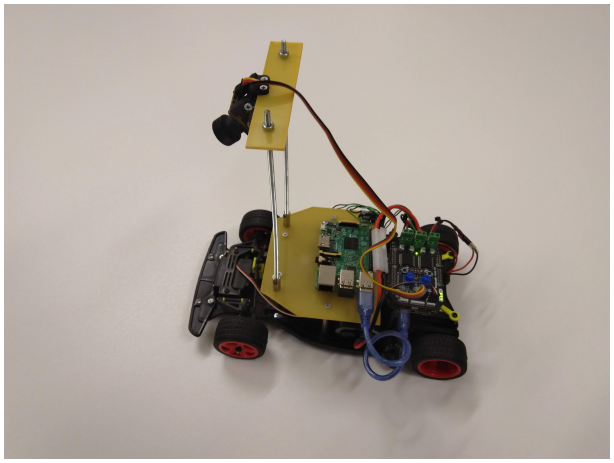
Stejně jako podvozek typu Alamak, Model C obsahuje servomotor pro ovládání natočení kol a řádkovou kameru.

Výška	70 mm
Šířka	160 mm
Délka	285 mm
Rozvor kol	160 mm
Rozchod kol	200 mm
Průměr kol	50 mm
Pohon	2x DC motor třídy 260, 7.2V
Maximální natočení kol	40°

Tabulka 3: Technické parametry podvozku typu Model C

3.2.2 FRDM-KL25Z

FRDM-KL25Z je velmi levná vývojová platforma firmy NXP pro MCU řady Kinetis KL14, KL15, KL24 a KL25. Platforma je řízena mikrokontrolerem Arm® Cortex®-M0+, konkrétně



Obrázek 10: Auto s podvozkem typu Model C

MKL25Z128VLK4 MCU s taktem 48 MHz, 128 KB flash paměti a 16KB SRAM paměti. Pro připojení slouží dva Mini-B USB konektory, jeden pro programování a jeden pro uživatelskou funkčnost. Vývojová platforma má vyvedených 54 GPIO pinů, které jsou Arduino kompatibilní. Deska také obsahuje kapacitní dotykový "slider"a akcelerometr `MMA8451Q`. Pro programování a ladění slouží OpenSDA rozhraní, případně nepřipájené SWD piny pro připojení ladicího rozhraní JLink. Pro uživatelský výstup slouží také RGB LED dioda. [4]

3.2.3 TFC Shield

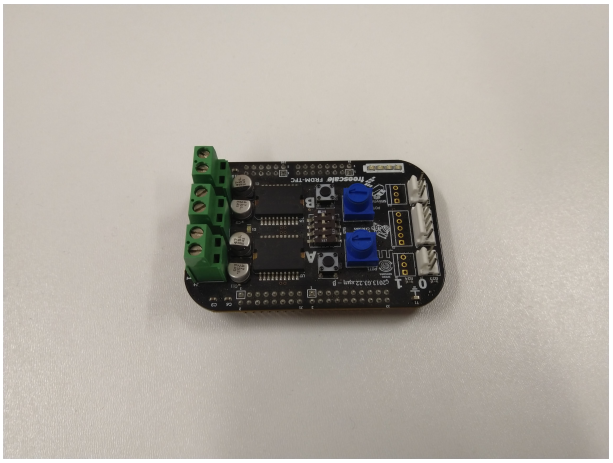
TFC shield je rozšiřující deska pro desku FRDM-KL25Z. TFC shield sjednocuje rozhraní pro připojení periferií k vývojové desce. Na shieldu nalezneme dva H-můstky `MC33887APVW` pro ovládání výkonu motorů, dva konektory pro připojení servomotorů, dvě rozhraní pro připojení řádkových kamer, dvě rozhraní pro připojení senzorů rychlosti, dva univerzální potenciometry, čtyři univerzální DIP přepínače a 4 univerzální LED diody. Funkčnost LED diod, DIP přepínačů a potenciometrů lze naprogramovat a můžou sloužit např. pro přepínání jednotlivých jízdních režimů, nebo pro úpravu parametrů řízení v případě přepínačů a potenciometrů; nebo k různé signalizaci a identifikaci v případě LED diod.

3.2.4 Raspberry Pi

Raspberry Pi je malý, levný, jednodeskový počítač o velikosti zhruba platební karty. První verze Raspberry Pi pochází z roku 2012. Při vypracování této práce byly používány dva typy:

1. Raspberry Pi 3 B
2. Raspberry Pi Zero W

Raspberry Pi 3B je součástí školního modelu auta s podvozkem typu Model C, Raspberry Pi Zero bylo použito při domácím testování. Raspberry Pi podporuje celou řadu operačních systémů, ať



Obrázek 11: Oficiální TFC shield

už linuxové, nebo nelinuxové. Při vypracování práce byla použita na obou modelech Raspberry Pi linuxová distribuce Raspbian. Srovnání parametrů obou verzí Raspberry Pi je vypsáno v Tabulce 4.

Raspberry Pi je připojeno k platformě FRDM-KL25Z pomocí USB rozhraní a komunikují mezi sebou sériově. Model auta je obohacen o Raspberry Pi z několika důvodů:

1. Raspberry Pi je mnohem výkonnější, než platforma FRDM-KL25Z a umožňuje např. složitější metody zpracování obrazu, či mnohem jednodušší a rychlejší nastavování parametrů (viz kapitola 5.2).

2. Raspberry Pi v obou použitých verzích obsahují vestavěné Wi-Fi rozhraní. Jelikož oproti FRDM-K66F neobsahuje deska FRDM-KL25Z ethernet rozhraní, je možné pro bezdrátovou komunikaci použít právě Raspberry Pi.

3. Raspberry Pi podporuje vzdálený přístup a lze se na něj připojit pomocí SSH. Je tak možno přeprogramovat zařízení vzdáleně. Na Raspberry Pi je zároveň spuštěna služba, která veškerý sériový přenos přeposílá pomocí UDP protokolu. Pokud je Raspberry Pi připojeno do školní sítě, dostane od DHCP adresu a je možno se pomocí VPN připojit i mimo školní síť.

	Pi 3 B	Pi Zero
CPU	Arm Cortex A53	ARM1176JZF-S
Takt	900 MHz	1 GHz
Jader	4	1
RAM	1GB	512 MB
Velikost	85.6 mm x 56.5 mm	65 mm x 30 mm

Tabulka 4: Porovnání verzí Raspberry Pi

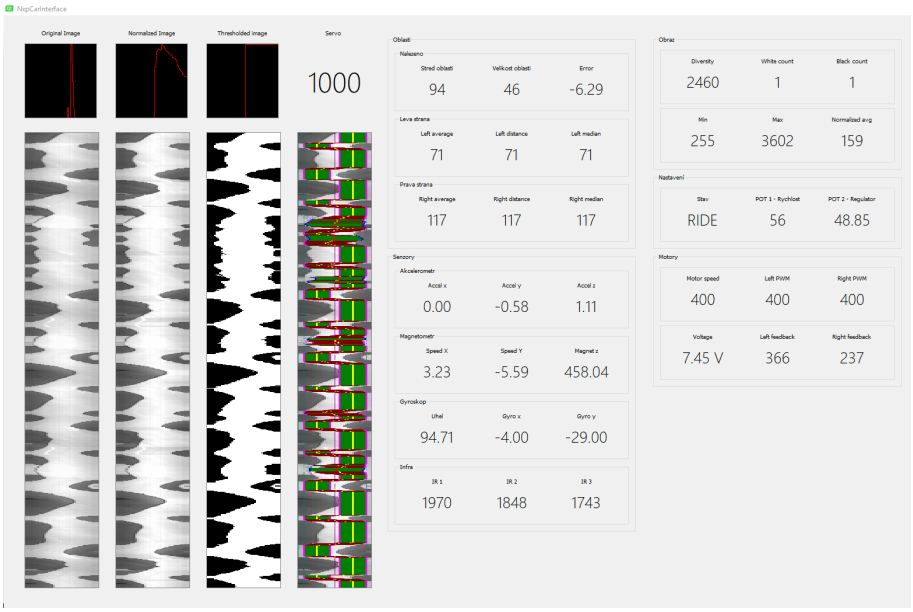
4 Podpůrný software

4.1 NXP Car Interface

Při vývoji softwaru pro autonomní vozidlo je velmi důležité disponovat nástrojem, který v reálném čase zobrazí veškerá důležitá data. Pro tyto účely byla vyvinuta aplikace NXP Car Interface. Nejdůležitější prvek aplikace je možnost zobrazit v reálném čase výstup z kamery v čitelném formátu - ve 2D formátu s historií záznamu (viz levá strana Obrázku 12). V aplikaci je možné zobrazit více postupných kroků ve zpracování obrazu. Je zobrazen tedy originální obraz, na který už je aplikován mediánový filtr, normalizovaný obraz, prahovaný obraz a zpracovaný obraz, ve kterém je zobrazen nalezený Region. Dále aplikace slouží pro zobrazení numerických hodnot pěti základních kategorií: Regiony, Senzory, Obraz, Motory a Nastavení (viz pravá strana Obrázku 12). Aplikace byla vytvořena v jazyce C++ s použitím knihoven Qt a OpenCV. Pro komunikaci s modelem auta byl zvolen protokol UDP - viz dále kapitola 5.1.

Pro optimalizaci objemu přenášených dat mezi modelem auta a desktopem se nepřenáší všechny tři stavy obrazu, ale je přenášen pouze surový, neupravený obraz. Obraz je dále zpracován desktopovou aplikací pomocí stejných algoritmů, které jsou využívány pro zpracování obrazu mikroprocesorem modelu auta. Obraz je přenášen už s aplikovaným mediánovým filtrem z důvodu ověření správnosti nastavení filtru.

Aplikace také ukládá záznam z jízdy do CSV souboru pro další analýzu dat. Do CSV souboru jsou ukládány veškeré numerické hodnoty zobrazené v pravé části aplikace.



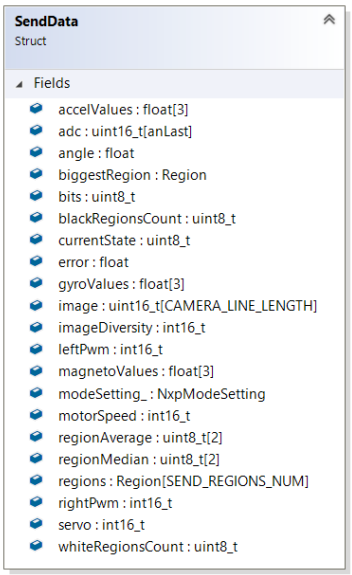
Obrázek 12: Aplikace NXP Car Interface

5 Řídící software

5.1 Bezdrátový přenos dat

Při přenosu dat je důležité, aby přenos probíhal asynchronně a aby nebyl spojově orientovaný, proto byl vybrán pro komunikaci s počítačem při jízdě protokol UDP. Vývojová deska se chová jako UDP server a aplikace NXP Car Interface se chová jako UDP klient. Veškerá komunikace probíhá přes broadcast na zvoleném portu.

Pro bezdrátový přenos dat slouží struktura `SendData`. Struktura pro přenos dat je jednoduše rozšiřitelná o jakékoliv parametry, které je nutné přenášet. V základu je přenášen nezpracovaný obraz z řádkové kamery (resp. obraz po aplikaci mediánového filtru), hodnoty z AD převodníků, hodnoty PWM pro motory a nastavení servomotoru. Dále jsou přenášeny veškerá nutná diagnostická data. Přehled přenášených dat je zobrazena na Obrázku 13



Obrázek 13: Diagram struktury `SendData`

5.2 Rozhraní pro změnu nastavení

Při experimentech s Raspberry Pi se ukázalo, že by bylo výhodné mít uložené nastavení různých řídících parametrů mimo kompilovaný program. Při nastavování konstant je zdlouhavé a nepraktické neustále rekompilovat aplikaci. Bylo tedy vytvořeno rozhraní pro načtení nastavení z uloženého dokumentu.

Veškeré nastavení konstant pro Raspberry Pi je uloženo v XML souboru, který je při spuštění souboru zpracován jednoduchým parsováním, které je zajištěno třídou `XmlReader`. Při spuštění programu se celý XML soubor načte do paměti do řetězce, parsování poté probíhá pouze jako rekurzivní hledání podřetězců dle zadaných parametrů. Jednoduché hledání podřetězce je zobra-

zeno ve Výpise 1. Jelikož je většina samotných konfiguračních dat ve formě numerických hodnot, musí je umět třída `XmlReader` zpracovat. Pro zpracování numerických hodnot slouží standardní funkce jazyka C `strtof` pro reálná čísla a `strtol` pro celá čísla. Systém zpracování XML souboru je velmi jednoduchý, ale jeho nevýhodou je absence ověřování vstupů a chyb. Zároveň nelze do XML souboru z programu zapisovat. Veškerá data z XML souboru jsou po celou dobu běhu programu načteny v paměti, nejsou získávána dynamicky podle potřeby, což by mohlo být bráno jako nevýhoda.

```
1  std::string XmlReader::getChildString(const std::string &xmlString, const std::string &name)
       {
2      std::string begName = "<" + name + ">";
3      const std::string endName = "</" + name + ">";
4
5      std::string xmlString = xmlString;
6      long long beg = xmlString.find(begName);
7      if (beg < 0)
8          return "";
9      beg += std::strlen(begName.data());
10
11     const long long end = xmlString.find(endName);
12     if (end < 0)
13         return "";
14     return xmlString.substr(beg, end - beg);
15 }
```

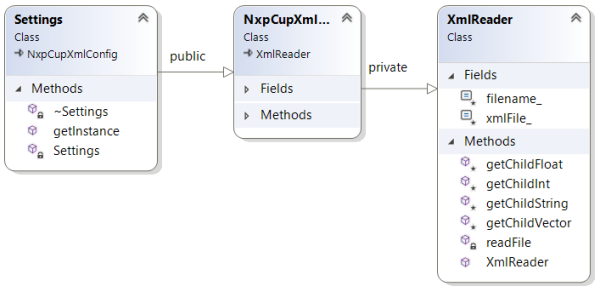
Výpis 1: Jednoduchý algoritmus získání prvku v XML souboru

Se třídou `XmlReader` pracuje dále třída `NxpCupXmlConfig`, která z ní dědí a která slouží v podstatě pouze jako prostředník pro data z XML souboru. Třída `NxpCupXmlConfig` uchovává v konstantních proměnných veškeré parametry z XML souboru a poskytuje metody pro získání těchto konstant (tzv. "gettery"). Ze třídy `NxpCupXmlConfig` následně dědí třída `Settings`, která je implementována jako návrhový vzor Singleton, což v zaručuje jedinou instanci této třídy v celém programu. Díky použití návrhového vzoru Singleton lze přistoupit ve kterékoliv části kódu k nastavení z XML souboru.

Původní platforma `FRDM-KL25Z` nedisponuje jednoduchou možností práce se soubory na vyměnitelném úložišti. Nová platforma `FRDM-K66F` ale disponuje rozhraním pro připojení paměťové karty typu Micro SD. Práce se soubory na vložené kartě je triviální, ale z důvodu nedostatků systému zpracování XML souboru nebylo pro platformu `FRDM-K66F` rozhraní implementováno.

5.3 Abstrakce tříd

Cílem bakalářské práce bylo vytvořit software, který lze zkompilovat beze změn jak na vývojových deskách FRDM, tak pro platformu Raspberry Pi. Jediný rozdíl mezi softwarem pro



Obrázek 14: Diagram dědičnosti tříd

Raspberry Pi a pro vývojové desky FRDM je ve způsobu získání dat z třídy TFC. Jelikož je k Raspberry Pi připojena deska FRDM-KL25Z pomocí USB rozhraní a komunikují sériově, rozhraním UART, je potřeba postarat se o příjem a odeslání dat. Samotná komunikace ovlivní pouze část kódu, zbylá část aplikace je zkompilovatelná beze změn na všechna zařízení.

Zároveň je potřeba postarat se o přenos dat pro aplikaci NXP Car Interface. Bezdrátovému přenosu dat je více věnována kapitola 5.1. V ideálním případě je možné odesílat bezdrátová data jednotně, bez nutnosti velkých změn. Platformy FRDM-K64F a FRDM-K66F disponují rozhraním Ethernet a pro odesílání dat existuje specializovaná třída `Enet`. Raspberry Pi obsahuje vestavěný Wi-Fi modul a ze softwarového hlediska je potřeba pracovat pomocí Socketů. Vývojová platforma FRDM-KL25Z neobsahuje ani Wi-Fi modul, ani ethernet rozhraní. Není tedy možné nativně komunikovat pomocí Wi-Fi.

Pro jednotný přístup byly vytvořeny abstraktní třídy, které byly následně implementovány pro každou platformu zvlášť.

5.4 Požadavky

Řídící software musí být navržen takovým způsobem, aby mohl být beze změn zkompilován nejen pro Raspberry Pi, ale i pro mikropočítač FRDM-K66F. Při tvorbě softwaru byly využity principy OOP a návrhových vzorů. Řídící software se skládá ze dvou hlavních částí: zpracování obrazu z řádkové kamery a řízení podle získaných informací z obrazu.

5.5 Zpracování obrazu

Zpracování obrazu obecně je disciplína náročná na výpočetní výkon. Většinou se zpracováním obrazu myslí práce s obrazovými daty ve 2D prostoru – obrazová data jsou uložena v matici. Za takový obraz můžeme považovat například fotografii. Řádková kamera nám ale poskytuje obrazová data v 1D prostoru – získáváme data v rozlišení 1x128 bodů. Taková obrazová data se nedají smysluplně zobrazit stejným způsobem, jako konvenční 2D obraz. Pro zobrazení dat z řádkové kamery máme v základu dvě možnosti. Jelikož k obrazovým datům z řádkové kamery můžeme přistupovat jako k numerickým hodnotám, můžeme data zobrazit v grafu, nebo na osciloskopu. Jelikož se ale pořád jedná o obrazová data, můžeme je zobrazit ve větším kontextu

jako konvenční obraz – tzn. nezobrazujeme jen jeden řádek, ale zobrazujeme větší počet řádků najednou. Tímto přístupem můžeme efektivně vizualizovat historii záznamu (viz kapitola 12). Zpracování obrazu v této konkrétní aplikaci probíhá v následujících krocích:

1. Získání obrazu z kamery
2. Filtrování mediánem
3. Normalizace obrazu
4. Prahování průměrem

5.5.1 Kalibrace kamery

Pro správnou funkčnost systému je potřeba, aby kamera byla správně zkalibrována. Obraz z kamery je ovlivněn následujícími parametry:

1. Zaostření kamery
2. Natočení kamery
3. Náklon kamery
4. Výška kamery

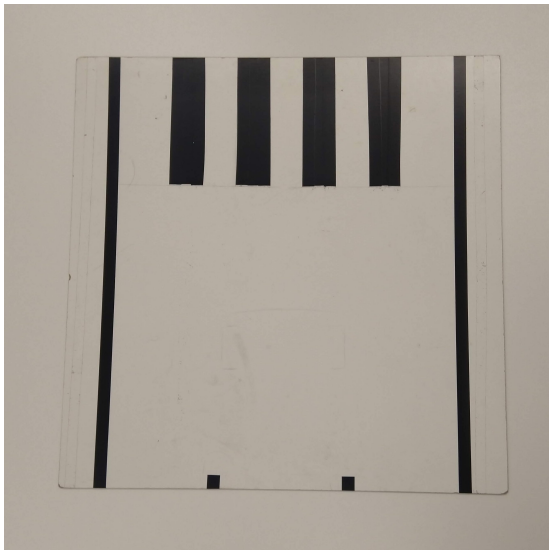
Pro kalibraci kamery byla vytvořena jednoduchá kalibrační deska (viz Obrázek 15), pomocí které je možné nastavit zaostření a natočení kamery. Kalibrační deska má vyznačené čáry, které se svou velikostí a vzdáleností shodují s čárami oficiální dráhy. Zároveň je pro potřeby kalibrace vyznačena středová pozice auta na dráze. Dále je také na desce pro přesnější kalibraci zaostření vyobrazen vzor, shodující se se segmentem dráhy Začátek zóny zpomalení (viz kapitola 2.2.3)

Nejprve je potřeba nastavit zaostření kamery. Pro zaostření kamery je potřeba položit auto na podložku podle Obrázku 16. Ostření kamery probíhá šroubováním objektivu do závitu kamery. Kameru je potřeba zaostřit takovým způsobem, aby byly na prahovaném obraze viditelné všechny čáry. Správně zaostřený výstup je zobrazen na Obrázku 17.

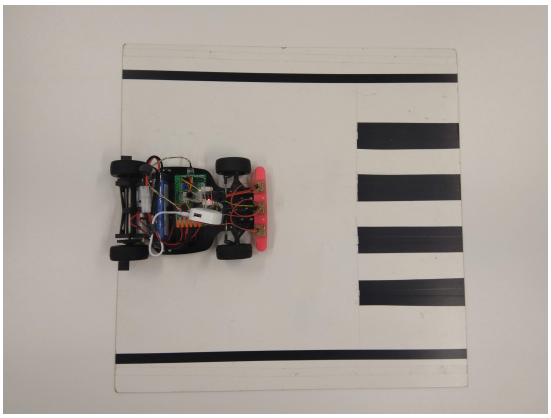
Pozn.: v aplikaci NxpCarInterface je obraz z kamery zobrazován po aplikaci mediánového filtru. Ve skutečnosti je obraz ostřejší. Pro potřeby správného zaostření je potřeba se řídit hlavně podle prahovaného obrazu.

Po správném zkalibrování kamery je potřeba kalibrovat také natočení kamery. Kamera by měla být vždy natočena na střed obrazu. Pokud nebude ve výchozím nastavení kamera natočena přesně na střed obrazu, bude regulace auta náchylná k chybám.

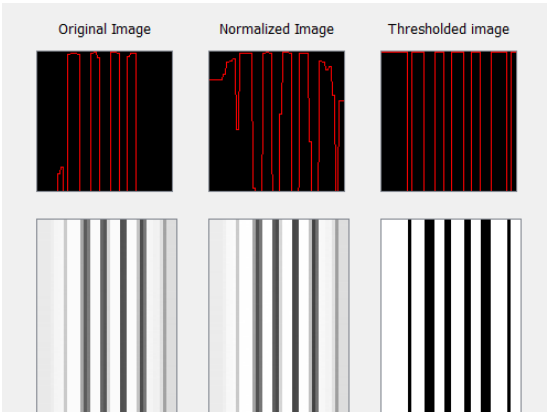
Pro kalibraci je potřeba položit auto před kalibrační podložku tak, aby se přední kola dotýkaly okraje podložky ve vyznačených místech. Dále je potřeba zkontrolovat, že všechna kola auta jsou rovnoběžně s čárami na podložce. V takové pozici je auto na středu podložky (viz Obrázek 18). Poté je potřeba otáčet kamerou tak, aby byla chyba zobrazená v aplikaci NxpCarInterface co nejnižší, ideálně pak nulová (nastavované hodnoty viz Obrázek 19).



Obrázek 15: Kalibrační deska



Obrázek 16: Kalibrace zaostření kamery



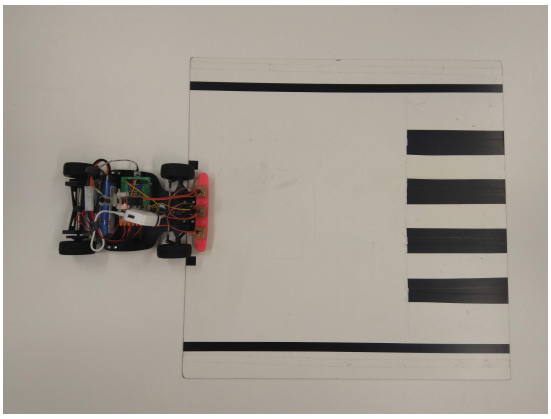
Obrázek 17: Vizualizace zaostřené kamery v aplikaci NxpCarInterface

5.5.2 Získání obrazu z kamery

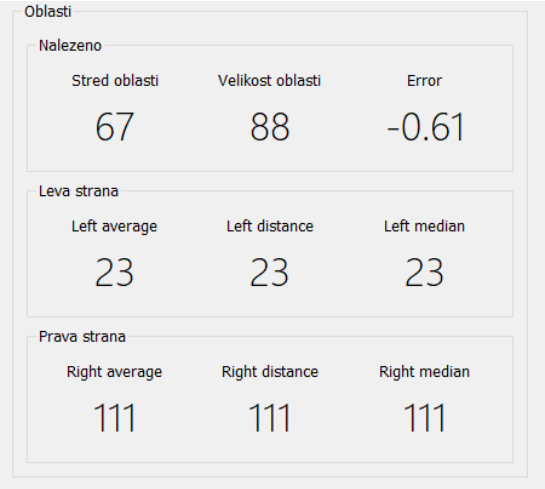
Získání obrazu je zapouzdřeno třídou TFC, v programu se dále pracuje pouze s 1D polem datového typu `uint16_t` (16 bit celočíselný typ bez znaménka). Hodnota jednotlivých bodů se nachází v intervalu $< 0, 4095 >$.

5.5.3 Filtrování mediánem

Originální obraz z kamery může být ovlivněn šumem, nebo např. nepořádkem na trati. Pro robustní rozpoznávání je potřeba obraz vyhladit. Pro vyhlazování mohou být použity různé filtry jako například: bilineární filtr, Gaussovo filtrování, nebo filtrování mediánem. V této konkrétní aplikaci bylo použito právě filtrování mediánem (viz výpis 2). Mediánový filtr dosahuje dobrých



Obrázek 18: Kalibrace natočení kamery



Obrázek 19: Hodnoty při kalibraci natočení kamery v aplikaci NxpCarInterface

výsledků odstranění šumu a zároveň nedeformuje hrany, na rozdíl od Gaussova bilineárního filtr. Často se tedy používá pro zpracování obrazu umělou inteligencí.

Pro aplikaci mediánového filtru je potřeba iterovat obrazem a uložit si bod a jeho okolí. Z bodu a jeho okolí se vypočítá medián, kterým je nahrazen daný bod.

```
1  //NxpImageAbstract.cpp
2  void NxpImageAbstract::slowMedianBlur(
3          const uint16_t (&srcImg)[CAMERA_LINE_LENGTH],
4          uint16_t (&dstImg)[CAMERA_LINE_LENGTH],
5          const int pixels) {
6      std::memcpy(dstImg, srcImg, CAMERA_LINE_LENGTH);
7      std::vector<uint16_t> blurBuffer;
8
9      for (int i = pixels; i < CAMERA_LINE_LENGTH - pixels; i++) {
10         for (int j = -pixels; j <= pixels; j++) {
11             blurBuffer.emplace_back(srcImg[i - j]);
12         }
13         std::sort(blurBuffer.begin(), blurBuffer.end());
14         dstImg[i] = blurBuffer.at(pixels + 1);
15         blurBuffer.clear();
16     }
17 }
```

Výpis 2: Algoritmus filtrování mediánem

5.5.4 Normalizace obrazu

V aplikacích strojového vidění jsou velmi důležité světelné podmínky. Při testování se ukázalo, že měnící se světelné podmínky na dráze jsou častý důvod ztráty orientace a vyjetí z dráhy.

Tento problém byl poměrně efektivně vyřešen normalizací obrazu. Zároveň normalizace obrazu řeší částečně i problém nedostatku světla. Implementaci algoritmu lze nalézt na výpisu 3.

Pro normalizaci obrazu je potřeba najít hodnotu minimálního a maximálního bodu v celém obraze. Poté je potřeba každý jednotlivý bod normalizovat z intervalu $< min_f, max_f >$ do intervalu $< 0, 255 >$, kde min_f a max_f jsou nalezené hodnoty minima a maxima. Výsledná normalizace zároveň konvertuje obraz z intervalu $< 0, 4095 >$ do intervalu $< 0, 255 >$. Díky této konverzi je ušetřen jeden byte pro každý bod za cenu snížené kvality obrazu. Praktický test ukázal, že snížení kvality obrazu nemá negativní účinky na rozpoznávání čar. Původní záměr pro snížení kvality bylo použití histogramu pro prahování, ale tento algoritmus nebyl ve finální aplikaci využit.

Díky normalizaci obrazu je auto schopné přizpůsobit se proměnným světelným podmínkám. Při samotném závodě jsou zajištěny konstantní světelné podmínky, ale při testování se ukázalo, že je vhodné, aby bylo zpracování obrazu přizpůsobivé změnám okolního osvětlení. Na Obrázku 20 lze vidět experiment, kde auto jezdilo v malém okruhu při špatných světelných podmínkách a poté byly skokově světelné podmínky zlepšeny. Auto objíždělo okruh spolehlivě jak při zhoršených světelných podmínkách, tak při ideálních světelných podmínkách. Zároveň se systém auta ihned adaptoval na skokovou změnu osvětlení a dále pokračoval ve spolehlivém řízení. Ze záznamu obrazu je také vidět, že zpracované Regiony a nalezené čáry jsou podobné jak při špatných světelných podmínkách, tak při optimálních podmínkách. Zároveň je také na záznamu vidět, že při zhoršených světelných podmínkách je zpracovaný obraz více zašuměný než při optimálních podmínkách, což ale nemá negativní vliv na regulaci pozice auta.

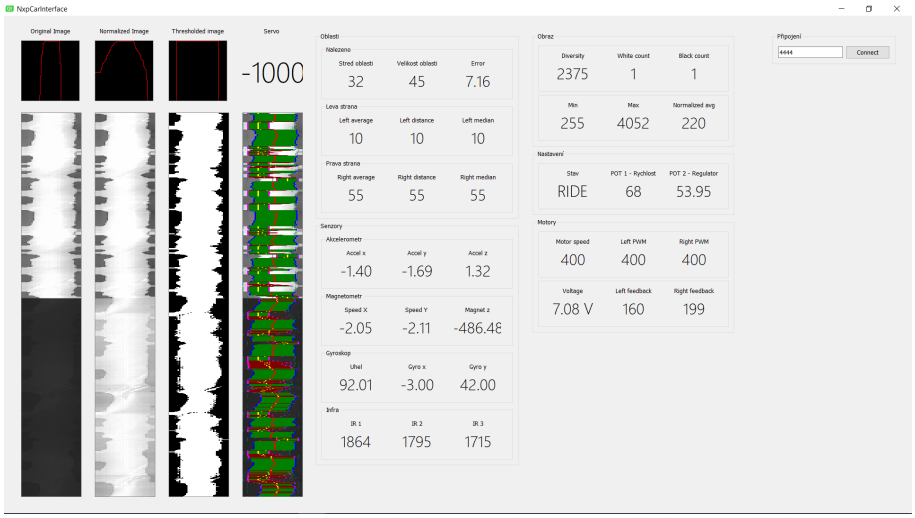
```
1  //NxpImageAbstract.cpp
2  void NxpImageAbstract::normalize(
3          const uint16_t (&srcImg)[CAMERA_LINE_LENGTH],
4          uint16_t (&dstImg)[CAMERA_LINE_LENGTH]) {
5      for (int i = 0; i < CAMERA_LINE_LENGTH; i++) {
6          float pixel = static_cast<float>(srcImg[i]);
7          pixel -= min_;
8          pixel *= COLOR_WHITE;
9          pixel /= (max_ - min_);
10         dstImg[i] = static_cast<uint16_t >(pixel);
11     }
12 }
```

Výpis 3: Algoritmus normalizace obrazu

5.5.5 Prahování průměrem

Posledním krokem při zpracování obrazu je prahování.

Prahování je převod hodnot do množiny o nižším počtu prvků, než měla původní data, zpravidla do množiny $\{0, 1\}$ [15].



Obrázek 20: Ukázka dynamické změny světlosti normalizací obrazu

Prahování se provádí na základě rovnice 1 [3].

$$g(x,y) = \begin{Bmatrix} 1, & f(x,y) \geq t \\ 0, & jinak \end{Bmatrix} \tag{1}$$

V této konkrétní aplikaci se převádí normalizovaný obraz v intervalu $< 0, 255 >$ do množiny $\{0, 255\}$, kde hodnota 0 je brána jako černá barva a 255 je brána jako bílá barva. Důležitým faktorem pro prahování je získání vhodného parametru t, který je poté použit v prahovací funkci. Existuje velké množství pokročilých přístupů k tomuto problému jako je například použití histogramu, shlukování nebo entropie. V této aplikaci byl zvolen jednodušší přístup prahování průměrem.

Pro prahování průměrem je potřeba najít průměrnou hodnotu bodu v obraze. Průměrná hodnota bodu je poté použita jako prahovací hodnota. Následně se provede samotné prahování, kde se každý bod nahradí hodnotou 0 je-li menší než průměrná hodnota, nebo hodnotou 255 je-li větší než průměrná hodnota.

5.6 Řízení pomocí dat z kamery

Pro samotné řízení je potřeba pracovat se zpracovaným obrazem. Základem řídícího systému je regulátor, který se stará o výpočet natočení kol. Vstupem do regulátoru je poté informace získaná pomocí nalezených čar.

5.6.1 Hledání čar

Pro hledání čar bylo otestováno a naprogramováno několik algoritmů.

První algoritmus hledal čáry od středu obrazu. Tento přístup se ukázal jako nevhodný, jelikož neřešil problém, kdy se jedna z čar objevila za středem. Tento problém se klasicky objevoval v zatáčkách, nebo u kraje dráhy.

Druhý algoritmus hledal čáry od krajů obrazu směrem ke středu. Tento přístup se ukázal jako vhodnější, ale neřešil problém, kdy se na trati objevila např. překážka, nebo stín, který tvořil pomyslnou čáru.

Třetí algoritmus (viz. Výpis 4) hledá v obraze tzv. Regiony. Region je v podstatě struktura, která uchovává indexy okrajů oblastí jedné barvy a danou barvu. Regiony jsou vyhledávány v obraze pomocí jednoduché smyčky, která si pamatuje aktuální barvu a porovnává ji s barvou bodu. Pokud bod nemá stejnou barvu, je nalezen přechod, a tudíž konec předchozího Regionu a začátek nového Regionu. Zároveň je aktualizována aktuální barva. S tímto přístupem v podstatě hledáme největší Region bílé barvy dané velikosti, který je obklopen z obou stran regionem černé barvy dané velikosti. Díky tomuto přístupu je možné jednoduše zjistit velikost a střed jednotlivých regionů, celkový počet Regionů v obraze nebo počet bílých a černých Regionů. Tímto přístupem je také částečně vyřešen implementační problém rozpoznání zón zpomalení a zrychlení.

```
1  //LineTracer.cpp
2  std::vector<Region> LineTracer::getRegions(const NxpImage &image, uint8_t searchLeftIdx,
       uint8_t searchRightIdx) {
3      uint8_t currentColor = static_cast<uint8_t>(image.atThresh(searchLeftIdx));
4      currentRegions_.emplace_back(Region({searchLeftIdx, searchLeftIdx, currentColor}));
5      for (uint8_t i = searchLeftIdx; i <= searchRightIdx; i++) {
6          if (currentColor != image.atThresh(i)) {
7              if (currentRegions_.size() > MAX_REGIONS_COUNT) {
8                  break;
9              }
10             currentRegions_.at(currentRegions_.size() - 1).right = i;
11             currentRegions_.emplace_back(Region({i, i, image.atThresh(i)}));
12         }
13         currentColor = static_cast<uint8_t>(image.atThresh(i));
14     }
15     return currentRegions_;
16 }
```

Výpis 4: Algoritmus hledání Regionů

Hledání čar pomocí Regionů je efektivní, ale pro určité případy je toto řešení příliš složité. Například jede-li auto rovně po rovině, obraz a vzdálenosti čar se v podstatě nemění. Není tedy potřeba vždy procházet celý obraz. Pro zjednodušení vyhledání čar byl vytvořen další, podpůrný algoritmus.

Abychom mohli zjednodušit vyhledávání čar, můžeme nejprve zjišťovat, jestli se v současném snímku nevyskytují čáry v podobných oblastech, jako v předchozím snímku. Algoritmus tedy prohledává v novém snímku okolí předchozích čar a hledá v nich barevné přechody. Nalezne-li na

obou stranách přechody v daném okolí, nemusíme následně prohledávat snímek pomocí Regionů. Kombinací s algoritmem pro hledání čar docílíme spolehlivého vyřešení problému rozpoznání zón zpomalení a zrychlení. Na druhou stranu může kvůli použití tohoto algoritmu zaostávat rozpoznávání překážky na trati. Pokud se objeví uprostřed trati překážka, auto ji bude ignorovat, pokud se nezměnily pozice čar.

V určitých případech se může stát, že v obraze není nalezena oblast, která by splňovala pravidla pro nalezenou oblast. Typicky to znamená, že největší bílý Region není z obou stran obklopen černým Regionem. S tímto problémem se můžeme setkat v zatáčce, nebo v křižovatce. Nachází-li se auto v zatáčce, typicky je nalezen pouze jeden černý Region, který obklopuje největší bílý Region. Tato situace nastává z důvodu, že řádková kamera při vjezdu do zatáčky ztratí informaci o vnitřní čáře. Je tedy nutné řídit se pouze pomocí vnější čáry. Pro optimální průjezd zatáčkou je vhodné zvolit stopu blíže vnitřnímu okraji, který není vidět. Je tedy potřeba auto virtuálně odsunout od vnější čáry zatáčky. V takovém případě je největší bílý Region upraven algoritmem 5. Vstup do algoritmu číslo 5 je ovlivněn předchozími přístupy hledání čar a díky tomu by nemělo dojít k chybnému určení čáry (např. levá čára by byla chybně identifikována jako pravá čára). Při vjezdu do zatáčky se při použití algoritmu 5 můžeme setkat s velmi náhlou změnou nalezených oblastí v závislosti na nastavení konstant `REGION_DISTANCE` a `REGION_COMPUTED_SIZE`. Velmi náhlá změna by se mohla teoreticky nepříznivě projevit na řízení auta, proto je vhodné při ztrátě jedné z čar uměle vytvořit postupný přechod okrajů Regionů od dříve nalezených k nově dopočítaným. Jedno z možných řešení je uchovávat si historii Regionů a upravovat dopočítaný Region pomocí průměrné, nebo mediánové hodnoty historie. Implementační problém přechodu okrajů Region je vyřešen a popsán v kapitole 5.6.3, kde je použit pro přechod medián.

```
1 //LineTracer.cpp
2 if (biggestWhiteRegion.getCenter() > CAMERA_LINE_LENGTH / 2) {
3     biggestWhiteRegion.left = biggestWhiteRegion.right - REGION_DISTANCE -
          REGION_COMPUTED_SIZE;
4     biggestWhiteRegion.right = biggestWhiteRegion.right;
5 } else {
6     biggestWhiteRegion.left = biggestWhiteRegion.left;
7     biggestWhiteRegion.right = biggestWhiteRegion.left + REGION_DISTANCE +
          REGION_COMPUTED_SIZE;
8 }
```

Výpis 5: Algoritmus dopočítání velikosti oblasti

5.6.2 Regulace pozice auta

Regulátor působí pomocí akční veličiny na soustavu tak, aby regulační odchylka byla co nejmenší. V tomto širším smyslu je regulátor složen z celé řady dalších částí. V praxi se nejčastěji používají

regulátory, které jsou složeny ze tří základních složek. Jedná se o proporcionální, integrační a derivační složku. Tím vznikají různé typy regulátorů až po PID regulátor. [9]

Pro ovládání řízení byl implementován PID regulátor, který pro svou proporcionální složku používá rozdíl poměrů vzdáleností čar od okrajů obrazu.

```
1 //NxpCarFreescale.cpp
2 const int leftDistance = left_;
3 const int rightDistance = CAMERA_LINE_LENGTH - right_;
4 const float leftRatio = static_cast<float>(leftDistance)/ static_cast<float>(rightDistance);
5 const float rightRatio = static_cast<float>(rightDistance)/ static_cast<float>(leftDistance)
      ;
6 float ratioDiff = rightRatio - leftRatio;
```

Výpis 6: Výpočet proporcionální složky pro PID regulátor

5.6.3 Regulace rychlosti

V jednodušším případě je možné nastavit oběma motorům konstantní rychlost. Toto řešení by bylo funkční, ale při vyšší nastavené rychlosti má auto tendenci vyjet z dráhy v zatáčce. Zároveň při vyšší rychlosti je potřeba mít v zatáčce agresivnější odezvu regulátoru. To znamená, že nastavená konstantní rychlost by musela být maximálně taková, při které auto ještě neopustí v zatáčce dráhu.

V této aplikaci byl implementován jednoduchý systém na ovládání rychlosti. Pokud je auto na rovince a zároveň jede rovně, může jet mnohem rychleji než v zatáčce, ve které musí zpomalit. Zároveň v zatáčce je potřeba použít diferenciál pro omezení vlivu nedotáčivého, nebo přetáčivého smyku.

Pro rozpoznání, zda je auto v zatáčce nebo na rovince slouží jednoduchý mechanizmus mediánu historie obrazu. Aplikace si uchovává historii zpracovaných záznamů (Regionů) a pro oba okraje je vypočítán medián. Pokud se okraje aktuálního řídícího Regionu nachází v blízkém okolí vypočteného mediánu, auto se pravděpodobně nachází na rovince a oba motory se mohou otáčet na maximální nastavený výkon. Pokud se jeden nebo oba okraje nenachází v okolí mediánu, nachází se auto pravděpodobně v zatáčce a je tedy potřeba zpomalit. Jelikož neznáme aktuální rychlost auta, je vhodné uložit si údaj, jak dlouho jede auto po rovině. Jede-li auto po rovině delší dobu, je potřeba při vjezdu do zatáčky brzdit mnohem větší silou, než pokud auto přijíždí do zatáčky z krátké roviny, nebo např. z jiné zatáčky. Rychlost vjezdu do zatáčky je také možné zjistit pomocí obrazových dat, kdy je derivace vzdáleností čar vyšší při vyšší rychlosti. Závislost rychlosti auta na derivaci vzdáleností je lineární, je tedy možné jednoduše implementovat brzdný systém.

Pokud je auto pravděpodobně v zatáčce, bere se v potaz natočení servomotoru a vypočítává se rychlost jednotlivých kol. Obecně řečeno se kolo, nacházející se blíže středu otáčení se

musí točit pomaleji než kolo vzdálenější středu otáčení. Rychlost vnějšího a vnitřního kola je vypočítána podle vzorce 2 [13], parametry pro výpočet diferenciálu jsou popsány v tabulce 5.

$$v_{o,i} = v[1 \pm c(Btan\theta)/2l] \tag{2}$$

v_o	rychlost vnějšího kola
v_i	rychlost vnitřního kola
v	celková rychlost
c	koeficient pro diferenciál
B	rozchod kol
θ	úhel natočení kol v radiánech
l	rozvor kol

Tabulka 5: Parametry pro výpočet diferenciálu

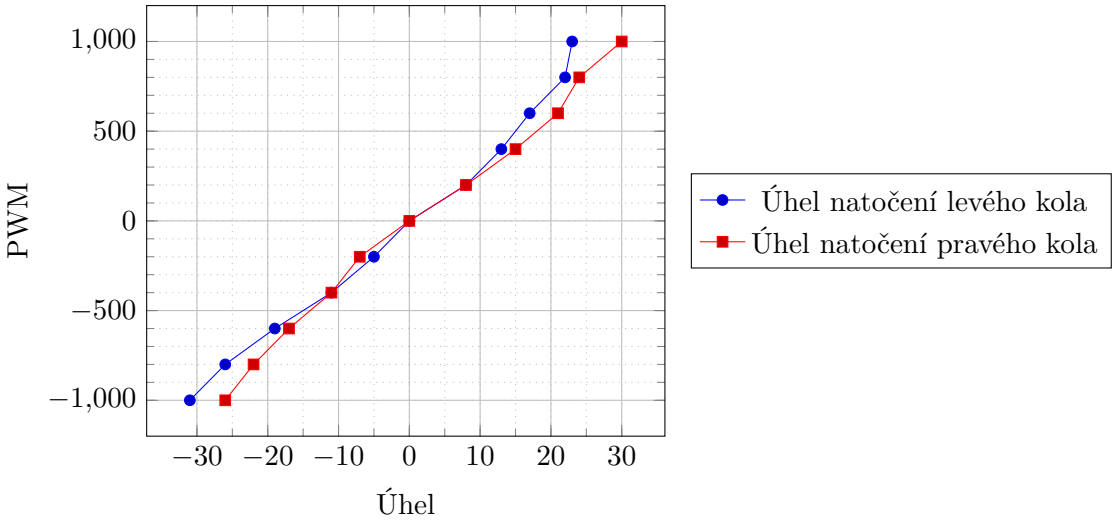
Standardně se pracuje s koeficientem $c = 1$, snižováním koeficientu lze dosáhnout nedotáčivosti, zvyšováním koeficientu lze dosáhnout přetáčivosti. Pro rychlejší průjezd zatáčkou byl experimentálně vybrán koeficient, při kterém auto projíždí zatáčkami při vyšší rychlosti mírným přetáčivým smykem.

5.7 Úhel natočení kol servomotorem

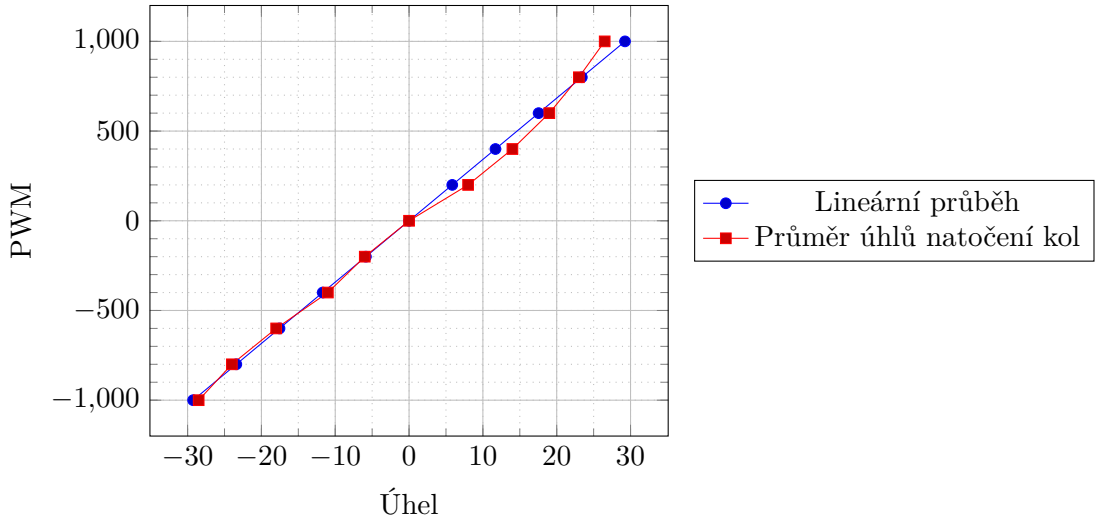
Pro výpočet diferenciálu je nutné pracovat s hodnotou natočení kol v radiánech - viz rovnice 2. Servomotor je ale ovládán pomocí PWM a není známa přesná hodnota natočení kol ve stupních. Pro aproximaci hodnoty natočení kol byl proveden experiment, kde byly postupně nastavovány hodnoty PWM pro servomotor z intervalu
$< -TFC\_SERVO\_MINMAX, TFC\_SERVO\_MINMAX >$ s krokem 200. Pro každou hodnotu PWM bylo poté změřeno natočení kol. Graf naměřených hodnot lze vidět na obrázku 21.

Z grafu naměřených hodnot lze zjistit, že natočení levého a pravého kola není stejné. Kola nejsou stejně natočena z důvodu použití tzv. Ackermannova podvozku. Na obrázku 22 lze vidět graf průměrné hodnoty natočení kol proložené lineární křivkou. Z grafu lze vyčíst, že průměrná hodnota v podstatě kopíruje lineární křivku a je tedy možné lineární závislost využít pro výpočet natočení kol v programu. Natočení kol je poté vypočítáno rovnicí 3, kde parametr p je roven PWM servomotoru. Jelikož rovnice 2 počítá s úhlem v radiánech, je třeba výsledek rovnice 3 převést do radiánů.

$$\begin{aligned} \theta_{deg} &= 5.85 \cdot p/200 \\ \theta_{rad} &= \theta_{deg} \cdot \pi/180 \end{aligned} \tag{3}$$



Obrázek 21: Naměřené hodnoty natočení kol



Obrázek 22: Průměr hodnot natočení kol proložený lineární křivkou

6 Zpracování dat senzorů

Pro spolehlivé řízení modelu auta se nelze spoléhat pouze na data z kamery. Přestože pro samotné řízení jsou data z kamery nejdůležitější, pro vylepšení jízdních vlastností je vhodné použít data z dodatečných senzorů.

6.1 IMU jednotka

V předchozích ročnících soutěže byl jeden z dílů na trati kopec, který bylo potřeba přejet. Pro řešení problematiky jízdy v kopci je vhodné použít data z gyroskopu, pomocí kterých lze jednoduše vypočítat náklon auta. V letošním ročníku závodu se kopec nenachází, není tedy třeba se zabývat danou problematikou.

Data z IMU jednotky ale mohou stále pomoci při řízení. IMU jednotka je v podstatě seskupení modulů pro měření polohy v prostoru – nejčastěji gyroskop, akcelerometr a magnetometr. Při testování byly otestovány následující IMU jednotky:

1. GY-521 MPU-6050
2. GY-955 BNO055
3. Vestavěná IMU jednotka na FRDM K66F

Při vypracování této práce se bohužel nepodařilo implementovat měření rychlosti, ani měření orientace pomocí dat z IMU jednotky (viz kapitola 7.3). IMU jednotka je však ve finální aplikaci použita pro detekci zastavení vozidla (viz kapitola 7.4).

6.1.1 Jednotka MPU-6050

Jednotka MPU-6050 na modulu GY-521 je velmi rozšířená, velmi levná šestiosá IMU jednotka. Narozdíl od dalších zmiňovaných jednotek nedisponuje magnetometrem, ale je výrazně levnější. Pro komunikaci slouží rozhraní I^2C. Na modulu nalezneme také DMP procesor, který se stará o výpočty nad naměřenými daty, a tak není potřeba vypočítávat jednotlivé úhly náklonu os.

6.1.2 Jednotka BNO055

Jednotka BNO055 na modulu GY-955 je levná a extrémně malá IMU jednotka. Udávaná přesnost zařízení výrobcem je až 0.01°, reálně je ale tato hodnota diskutabilní. Senzor BNO055 je vyráběn společností Bosch, nesplňuje tedy pravidla po použití v soutěži NXP Cup. Senzor se řadí mezi tzv. SMART senzory, což v praxi znamená, že uvnitř samotného čipu jsou umístěné komponenty pro zpracování signálu. Pro tento senzor to znamená např. to, že je v něm zabudován samostatný mikroprocesor ARM® Cortex®-M0, který se stará o zpracování a filtrování signálu. Mikroprocesor má nahraný firmware BSX3.0 FusionLib od společnosti Bosch práve pro zpracvání měřeného signálu. Teoreticky to znamená, že již nemusíme filtrovat signál ze senzoru

v hlavním řídícím mikropočítači. Použitím dodatečného mikropočítače na senzoru je porušeno další pravidlo pro NXP Cup.

GY-955 stejně jako IMU jednotka na K66F kombinuje gyroskop, akcelerometr a magnetometr. Jde tedy o devítiosý senzor.

6.1.3 IMU jednotka na FRDM-K66F

Na vývojové platformě FRDM-K66F nalezneme všechny tři zmíněné senzory, které tvoří devítiosou IMU jednotku. Konkrétně se jedná o akcelerometr a magnetometr *NXP FXOS8700CQ* a gyroskop *NXP FXAS21002* [6]. Oba zmíněné senzory komunikují s vývojovou deskou pomocí rozhraní I^2C. Podle pravidel NXP cup je nutné pro závody použít senzor od firmy NXP, pokud je takový senzor dostupný [10], z tohoto důvodu byly tyto senzory použity ve finální verzi programu.

6.2 Infračervený senzor

Pomocí dat z infračervených senzorů je možné spolehlivě zjistit, jestli auto přejelo cílovou čáru a zároveň, jestli auto vyjelo z dráhy. Hardware IR senzoru a zapojení bylo popsáno v kapitole 3.1.5. Z IR senzoru (resp. z kolektoru fototranzistoru) je měřeno napětí, které je poté ADC převodníkem na vývojové desce převedeno na digitální hodnotu. Pro jednoduché čtení z ADC převodníku byla upravena třída `TFC`, kde byl rozšířen existující kód čtení z ADC převodníku o nové kanály, na které je připojena trojice IR senzorů.

Data z infračervených senzorů musí projít prahováním, protože pro detekci přejetí čáry jsou důležité jenom dva stavy: stav kdy senzor detekuje černou barvu a stav, kdy není detekována černá barva. Teoreticky se na analogovém vstupu objevuje napětí jdoucí k 0V v případě, že není detekována černá barva a napětí blížící se referenčnímu napětí v případě detekce černé barvy. V praxi je odrazivost IR záření závislé na více faktorech - např. odrazivost materiálu. Prahovací hodnota byla určena empiricky jednoduchým pokusem, kdy bylo auto postaveno tak, aby všechny tři IR senzory zaznamenávaly černou čáru. Následně se nechalo auto projet dráhou (auto nesmí přejet žádnou černou dráhu) a byla nalezena maximální naměřená hodnota při průjezdu po čistě bílém povrchu. Prahovací hodnota je průměr těchto dvou hodnot.

Pro zjištění přejetí čáry je potřeba uchovat si určitou historii přejetí čáry. Přejede-li auto čáru, jen zřídkakdy na ni najede kolmo a jen zřídkakdy mají tedy všechny tři senzory jednotný výstup. Mechanismus historie přejetí čáry funguje tak, že jednotlivé informace ze senzorů jsou zakódovány do proměnné `lineCrossBits_`. Zároveň pro uchování historie je použita proměnná, která se při detekci černé oblasti nastaví na maximální hodnotu. V případě, že není detekována černá oblast se postupně v každém cyklu programu hodnota proměnné snižuje až na nulovou hodnotu. V případě nulové hodnoty proměnné je nastaven příslušný bit na hodnotu 0.

Zakódování všech tří senzorů do jedné proměnné má výhodu jednoduchého zjištění, jestli auto přejelo čáru - viz Výpis 7.

```
if (this->tfc_->ReadADC(anIR_1) > WHITE_IR_BOUND) {
    bitWrite(this->lineCrossBits_, 0, 1);
    this->leftLineCrossTimer_ = this->maxCrossTimer_;
} else if (this->leftLineCrossTimer_ > 0) {
    this->leftLineCrossTimer_--;
} else if (this->leftLineCrossTimer_ == 0) {
    bitWrite(this->lineCrossBits_, 0, 0);
}
```

Výpis 7: Ukázka zpracování vstupu IR senzoru

```
if (this->lineCrossBits_ == 0b111 || this->lineCrossBits_ == 0b101) {
    if (this->motorsState_ == MotorsState::Ride) {
        this->stop();
        }
    }
```

Výpis 8: Ukázka detekce přejetí čáry

7 Experimenty s Raspberry Pi

Jak již bylo řečeno dříve v kapitole 5.2, Raspberry pi je vhodné pro dynamické změny nastavení bez nutnosti rekompilace programu. Pouhou změnou parametrů v XML souboru lze jednoduše a pohodlně testovat nastavení.

7.1 Experimenty s nastavením PID regulátoru

Pro správné nastavení řízení je potřeba mít správně nastavené hodnoty koeficientů pro PID regulátor. Pro určení nastavení regulátoru je možné použít různé metody získání optimálního nastavení. Jedním z přístupů je použití tzv. Ziegler-Nicholsovy metody [8]. Ziegler-Nicholsova metoda získání parametrů je heuristická, to znamená že parametry nejsou získány matematickými výpočty nebo algoritmy, ale jsou získány experimentálně.

Popis experimentů

Principem Ziegler-Nicholsovy metody je postupné nastavování parametrů PID regulátoru. Nejprve je potřeba najít nastavení, ve kterém regulovaný systém dosahuje stabilní oscilace. Pro hledání takového nastavení je potřeba nastavit integrační konstantu K_i a derivační konstantu K_d na nulovou hodnotu a postupně zvyšovat proporcionální konstantu K_p. Jakmile regulovaný systém dosáhne stabilní a konzistentní oscilace, je možné pomocí této nalezené hodnoty K_u a pomocí periody oscilace T_u vypočítat hodnoty konstant proporcionální, derivační a integrační složky - viz Tabulka 6.

Název nastavení	K_p	K_i	K_d
Classic Ziegler-Nichols	0.6 K_u	0.5 T_u	0.125 T_u
Pessen Integral Rule	0.7 K_u	0.4 T_u	0.15 T_u
Some Overshoot	0.33 K_u	0.5 T_u	0.33 T_u
No Overshoot	0.2 K_u	0.5 T_u	0.33 T_u

Tabulka 6: Hodnoty pro parametry PID regulátoru [14]

Parametry pro PID regulátor je potřeba získat vždy při změně hardwaru auta. Např. auto s podvozkem Alamak potřebuje jinak nastavený regulátor, než auto s podvozkem typu Model C. Zároveň jsou hodnoty regulátoru závislé na vstupu obrazového signálu. V praxi to znamená, že např. při změně velikosti mediánového filtru, změně objektivu, změně zaostření nebo při změně citlivosti kamery je potřeba zkontrolovat správnost nastavení regulátoru.

Ziegler-Nicholsova metoda se ukázala jako spolehlivá, ale může nastat problém s měřením periody oscilace. Pro měření periody oscilace byl použit dlouhý segment rovné dráhy, kde model auta vždy začínal těsně u pravé čáry. Stabilita a konzistence oscilace lze velmi jednoduše odpozorovat, ale periodu je potřeba správně změřit. Ke změření byla použita existující vlastnost aplikace NXP Car Interface - ukládání dat do CSV souboru. Při analýze CSV souboru lze

pomocí hodnot servomotoru a pomocí známé frekvence odesílání dat jednoduše určit periodu oscilace.

Ziegler-Nicholsova metoda není vhodná pro určení hodnot pro regulátor při tréninkové části soutěže NXP Cup, jelikož není dostupný daný úsek dráhy. Je možné použít méně přesný postup, který by se dal shrnout do následujících kroků [12]:

1. Nastavení konstant K_p, K_i a K_d na nulovou hodnotu.
2. Postupně zvyšovat proporcionální konstantu K_p, dokud auto nedosáhne stabilní a konzistentní oscilace.
3. Postupně zvyšovat derivační konstantu K_d, dokud není oscilace eliminována.
4. Opakovat kroky 2 a 3, dokud zvyšování derivační konstanty K_d nemá vliv na eliminaci oscilace.
5. Nastavit konstantám K_p a K_d poslední známé hodnoty, ve kterých je ještě chování modelu auta stabilní.
6. Postupně zvyšovat konstantu K_i, dokud není nalezena optimální hodnota překmitů.

Při testování a v kvalifikačním kole závodu byly použity hodnoty pro PID regulátoru získané pomocí Ziegler-Nicholsovy metody, konkrétně tři mírně upravená nastavení "Some Overshoot". Pro finálové kolo byly poté použity nastavení získané druhým popsaným způsobem.

Výsledek experimentů

Výsledkem experimentů byly hodnoty koeficientů pro PID regulátor. Při experimentech se také ukázal význam jednotlivých složek regulátoru. Zvyšuje-li se proporcionální konstanta, model auta reaguje agresivněji a má tendenci se při vyšším nastavení rozkmitat. Derivační složka regulátoru ovlivňuje rychlost, se kterou je schopný regulátor reagovat na změnu. Integrační složka naopak rychlost reakce snižuje. Prakticky má integrační složka největší význam na zvlněné části trati a v křižovatce, kde sníží rychlost reakce a auto tak může vlnitou část trati projet mnohem rychleji.

7.2 Experimenty s nastavením diferenciálu

Při správném nastavení diferenciálu je možné vylepšit průjezd zatáčkami pomocí kontrolovaného přetáčivého smyku. V kapitole 5.6.3 je popsána funkčnost diferenciálu, kde jsou rychlosti pro jednotlivá kola vypočítána vzorcem 2. Jelikož ale není známa opravdová rychlost kol, ale pouze hodnota PWM pro jednotlivé motory, je dále pracováno pouze s hodnotou PWM.

Průběh experimentů

Experimenty probíhaly na dvou typech dráhy, s hodnotami pro regulátor, při kterých byl model auta v zatáčkách nedotáčivý. Prvním typem dráhy byl čtvercový ovál, druhým typem testovací dráhy byl stejný ovál, doplněný v jednom z rohů o křižovatku s krátkou technickou částí tvořenou větším množství zatáček. Postupným zvyšováním koeficientu c byla eliminována nedotáčivost. Při hladkém průjezdu zatáčkou je možné snížovat koeficient zpomalení v zatáčce a nadále zvyšovat koeficient diferenciálu. Ideální nastavení je takové, kdy auto projíždí zatáčkami mírným přetáčivým smykem s co nejmenší změnou rychlosti. Pokud je nastavený koeficient moc velký, auto má tendenci se přetočit příliš a může dojít k vyjetí z trati. V případě dvou zatáček za sebou ve tvaru písmene "U"může být přílišné přetočení žádoucí, v případě dvou zatáček za sebou ve tvaru písmene "S"je však přílišné přetočení nežádoucí a téměř vždy takové nastavení vede k vyjetí auta z dráhy. V extrémním případě příliš vysoké hodnoty koeficientu diferenciálu se stává, že model auta v zatáčce zastaví z důvodu vysoké brzdné síly vnitřního kola. Takové chování modelu auta je rovněž nežádoucí.

Výsledek experimentů

Experimenty s nastavením parametrů pro diferenciál ukázal, že správné nastavení velmi pomáhá při průjezdu zatáčkami. Při nalezení vhodného nastavení je poté vhodné znova nalézt hodnoty pro PID regulátor. Výsledkem experimentu je poté vhodná hodnota koeficientu c pro použití se vzorcem 2. Zároveň se při experimentu ukázalo, že absence senzoru rychlosti kol je nedostatek, který by bylo vhodné v budoucnu eliminovat.

7.3 Experimenty s IMU jednotkou

Původní myšlenkou pro použití IMU jednotky pro zlepšení vlastností bylo pouze průběžné měření rychlosti modelu auta. Při vypracování se ale naskytly další možnosti použití IMU jednotky, které jsou dále popsány v následujících kapitolách.

7.3.1 Měření rychlosti vozidla

Jelikož se při vypracování práce ukázalo, že by bylo vhodné znát aktuální rychlost modelu auta, bylo otestování měření rychlosti pomocí akcelerometru. Akcelerometr je senzor, který měří zrychlení ve třech osách x, y a z. Jelikož je zrychlení definováno jako derivace rychlosti podle času (vzorec 4), je možné spočítat zpětně rychlost pomocí integrace rychlosti (vzorec 5).

$$a = \frac{\Delta v}{\Delta t} \tag{4}$$

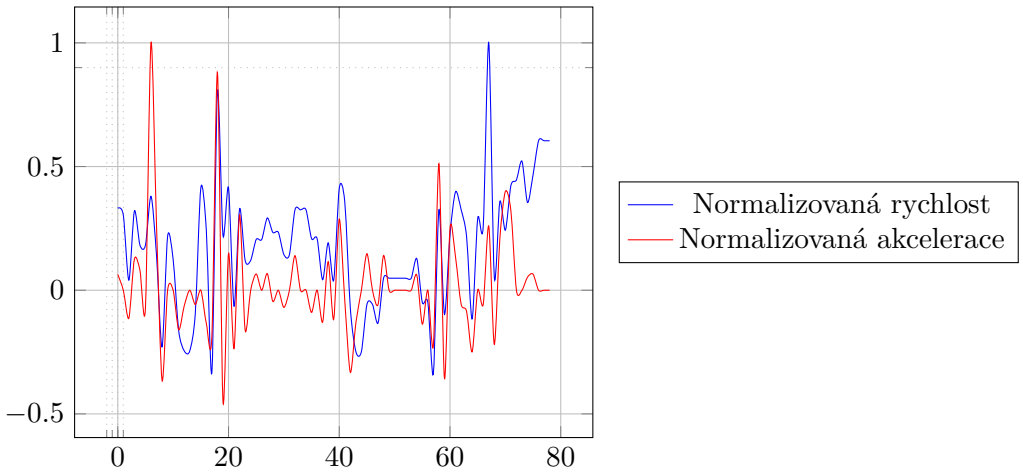
$$v = v + a \cdot \Delta t \tag{5}$$

Praktická implementace Při praktické implementaci se ukázalo, že data z akcelerometru jsou ovlivněny šumem a čistá data jsou tedy nepoužitelná. Proto je tedy potřeba data filtrovat. Filtrování dat probíhalo pomocí načtení většího množství hodnot do pole a následného průměrování, nebo mediánování hodnot. Oba filtry, průměrový i mediánový, měly pozitivní dopad na naměřená data a s takto upravenými daty bylo možné dále pracovat.

Dále se při experimentech ukázalo, že data z akcelerometru jsou ovlivněny gravitačním zrychlení v případě náklonu. Pomocí dat z gyroskopu je možné jednoduše odchylku vzniklou náklonem odstranit.

Následně se ukázalo, že data jsou i po předchozích korekcích ovlivněny jistou odchylkou, která dělá problém hlavně při klidovém režimu, kdy se s akcelerometrem nehýbe. Pro vyřešení se při startu programu provede automatická kalibrace, která si uloží určitý počet vzorků a stanoví medián a maximální naměřenou odchylku. Pro měření v aplikaci je poté odchylka používána jako hraniční hodnota pro horní propust. Zároveň je od naměřené hodnoty odečítán naměřený medián. Odečtením mediánu by měly být kompenzovány jakékoliv další odchylky. Od prahovací hodnoty horní propusti musí být medián také odečten.

Experimenty při jízdě probíhaly na malé čtvercové trati, kde auto objelo vždy jeden okruh. Po objetí jednoho okruhu a zastavení by měla být výsledná rychlost auta nulová. Nepodařilo se však takového cíle dosáhnout. Na Obrázku 23 je vidět graf zrychlení a vypočítané rychlosti v ose souběžné se směrem jízdy. Na konci jízdy by měla být rychlost nulová, ale nulová není. *Pozámka: Hodnoty v grafu jsou normalizovány do intervalu* $< 0, 1 >$



Obrázek 23: Ukázka dat z akcelerometru a spočítané rychlosti

Výsledek experimentů I přes všechny aplikované filtry na naměřené hodnoty podléhalo měření odchylkám. Přestože se podařilo kompenzovat odchylky v klidovém režimu, kdy je tedy možné pomocí dat z akcelerometru zjistit, jestli auto stojí, nepodařilo se vytvořit systém pro měření rychlosti jen za pomoci dat z IMU jednotky. Obecně se IMU jednotka nepovažuje za vhodný zdroj pro měření rychlosti a v reálných systémech jsou naměřená data vždy kombinována

s daty jiných senzorů (např. GPS, nebo samostatné senzory rychlosti). Jelikož auto nedisponuje žádným samostatným senzorem rychlosti, nebylo ani možné zpracovat hodnoty odchylek. Za předpokladu, že by IMU jednotka disponovala velikou přesností měření, by měření rychlosti bylo možné. Za předpokladu vysoká přesnosti by poté bylo možné měřit i aktuální polohu auta ve 3D prostoru. Ale s běžně dostupnými IMU jednotkami toto není možné, veškerá měření rychlosti jsou pouze orientační s velkou kumulativní chybou[11].

7.3.2 Měření orientace vozidla

IMU jednotky se ve velké míře používají v navigačních systémech mimo jiné z důvodu možnosti měření orientace. Jedná se tedy v podstatě o elektronický kompas. Experimenty s elektronickým kompasem měly za úkol vylepšit vlastnosti auta v různých segmentech trati a přidat nový parametr pro řízení auta. Vycházíme-li z předpokladu, že dráha je tvořena pouze díly, které mají celkově pouze tři směry, kam mohou směřovat (rovně, vlevo a vpravo) a zároveň všechny zatáčky jsou kolmé, můžeme nastavení vstupu regulátoru ovlivnit elektronickým kompasem.

Příklad použití Auto začíná na dráze na rovinném úseku a vjede do zatáčky. Systém regulace pozice auta zná aktuální natočení auta a je známo, že je potřeba se v zatáčce otočit přesně o 90°. PID regulátor řízení nepracuje pouze s proporcionální složkou pozice auta vůču čarám, ale pracuje také s aktuální orientací auta. Při rozeznání zatáčky je v závislosti na orientaci zatáčky nastaven cílový úhel natočení auta. Při správném nastavení regulátoru je možné mnohem přesněji projíždět zatáčkami a případně implementovat algoritmus pro určení ideální stopy zatáčkou. Při průjezdu zatáčky řádková kamera zachycuje zpravidla pouze jednu, vnější čáru. Díky dat z elektronického kompasu není složité dopočítat polohu vnitřní čáry.

Implementace Po načtení dat z IMU jednotky a následném upravení hodnot je potřeba vypočítat orientaci ve třech základních směrech, které jsou podle leteckého průmyslu pojmenovány jako klopení, klonění a bočení (anglické termíny: roll, pitch, yaw). Tyto vypočtené hodnoty jsou ovlivněny kumulativní chybou, tzv. driftem. Kumulativní chyba je v reálných navigačních systémech korigována např. pomocí systému GPS, lze ji ale korigovat i magnetometrem, který je na IMU jednotce k dispozici.

Nejprve je potřeba vypočítat hodnoty goniometrických funkcí *sin* a *cos* pro hodnotu úhlu klopení.

$$\begin{aligned} \alpha &= arctg2(a_y, a_z) \\ s &= sin(r) \\ c &= cos(r) \end{aligned} \tag{6}$$

Následně je potřeba korigovat rotaci zařízení podle následujícího vzorce:

$$\begin{aligned} y &= y_m \cdot c - z_m \cdot s \\ z_m &= z_m \cdot c + y_m \cdot s \\ z_a &= y_a \cdot s + z_a \cdot c \end{aligned} \tag{7}$$

Poté je vypočítán úhel klonění a hodnoty goniometrických funkcí *sin* a *cos* pro tento úhel:

$$\begin{aligned} \beta &= arctg2(-x_a, z_a) \\ s &= sin(p) \\ c &= cos(p) \end{aligned} \tag{8}$$

Posledním krokem je odstranění rotace magnetometru a výpočet úhlu bočení - tzn výpočet úhlu orientace vozidla:

$$\begin{aligned} x &= m_x \cdot c + m_z \cdot s \\ \gamma &= arctg2(-y, x) \end{aligned} \tag{9}$$

Poznámka: Vzorce byly zpracovány za pomocí zdrojů [1] a [2]

Výsledek experimentů Výsledkem implementace je po softwarové stránce funkční algoritmus elektronického kompasu. Při reálném testování se ale ukázalo, že hodnoty magnetometru jsou silně ovlivněny elektromotory modelu auta a nelze tedy tento senzor použít v umístění, ve kterém se standardně nachází. Řešením tohoto problému by mohlo být pořízení samostatného senzoru namísto vestavěného a připevnit jej nad kameru, kde by neměl být magnetometr ovlivněn. V případě umístění senzoru nad kameru by naměřená data při jízdě mohly být silně ovlivněny rozkmitáním tyče držící kameru a bylo by potřeba naměřené hodnoty za senzorů dále zpracovat.

7.4 Detekce zastavení vozidla

Systém řízení modelu auta pracuje ve čtyřech základních módech:

1. Stání
2. Rozjezd
3. Jízda
4. Zastavování

Při módu stání není aktivní servomotor a nepracuje tedy ani PID regulátor. Původní koncepce programu byla pouze třístavová, kde nebyl implementován mód Zastavování. Při testování se ukázalo, že pokud je cílová čára těsně před zatáčkou, auto má při vyšší rychlosti tendenci po

přejetí cílové čáry při zastavování vyjet z dráhy z důvodu nečinnosti PID regulátoru. Při závodech NXP Cup je vyjetí z dráhy po úspěšném dokončení kola penalizováno ztrátou jedné vteřiny. Proto byl implementován mód Zastavování. Pro mód Zastavování je vhodné, aby byl systém adaptivní. V praxi to znamená, že při vyšších rychlostech potřebuje model auta delší čas a delší dráhu na zastavení, než při nižší rychlosti. Adaptivitu algoritmu zajišťují data z IMU jednotky, konkrétně data z akcelerometru. Principem měření zastavení je sledování akcelerace auta, která bude po samotném zastavení konstantní, v případě nezatížení senzoru chybou bude hodnota akcelerace nulová. Zastavení je poté realizováno ve dvou fázích. První fáze je samotné zpomalení, kdy je motorům nastavena záporná hodnota PWM pro rychlé zastavení. Jakmile data z akcelerometru přesáhnou nastavenou mez, je motorům nastavena nulová hodnota PWM a stav se přepne do módu Stání poté, co se hodnoty z akcelerometru ustálí.

8 Účast na NXP Cup

Vyvinutý software pro model auta byl testován na soutěži NXP Cup v regionálním kole. Po regionálním kole byl software dále upraven a vylepšen a finální verze byla použita pro finální kolo.

8.1 Regionální kolo

Regionální kolo soutěže NXP Cup se odehrálo 21.3.2019 na půdě Fakulty elektrotechniky a informatiky VŠB-TUO. Do regionálního kola bylo přihlášeno celkem 26 středoškolských a vysokoškolských týmů z České republiky, Polska, Slovenska a Ukrajiny. Z těchto 26 týmů dosáhlo celkem 11 týmů nenulového hodnocení. Čtyřem týmům se podařilo dokončit jízdu v "osmičce", přičemž tři týmy sdílely shodný výsledek. Překážce se podařilo vyhnout 6 týmů a disciplínu omezení rychlosti úspěšně zvládlo 5 týmů. Hlavní závod poté úspěšně dokončilo 5 týmů. Kompletní tabulka výsledků je zobrazena v Tabulce 7. Přestože se týmu Richard's Gear, který soutěžil s popisovaným softwarem, nepodařilo dokončit závod, povedlo se týmu postoupit do finálního kola díky bodům získaným z dodatečných disciplín.

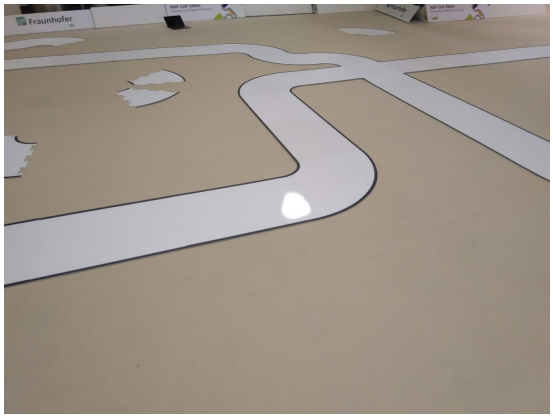
8.2 Finální kolo

Finální kolo soutěže NXP Cup se odehrálo 29.4.2019 - 30.4.2019 na půdě Fraunhoferského institutu pro integrované obvody (Fraunhofer-Institut für Integrierte Schaltungen). Finálního kola se zúčastnilo celkem 19 týmů z celého EMEA regionu, celkem z 11 zemí. Finální kolo bylo rozděleno do dvou dnů, kdy první den a polovina druhého dne sloužily pro trénink. Dráhy pro dodatečné disciplíny byly sestaveny stejně, jako ve kvalifikačním kole. Velký rozdíl mezi kvalifikačním kolem a finálním kolem byly světelné podmínky. Zatímco ve kvalifikačním kole byly zajištěny konstantní podmínky, v finálním kole byly světelné podmínky na dráze velmi proměnlivé. Některé části dráhy byly silně přesvětlené a kamera měla problém zachytit použitelný obraz. Zároveň se na dráze občas objevovaly silné odlesky (viz Obrázek 24), které měly negativní vliv na zpracování obrazu.

Z celkových 19 týmů celkem 13 týmů dosáhlo nenulového hodnocení (viz podrobněji v Tabulce 8). Tým Richard's Gear reprezentující VŠB-TUO se bohužel umístil mezi šesti týmy s nulovým hodnocením. Hlavní příčinou neúspěchu byly hlavně špatné a proměnlivé světelné podmínky.

Název týmu	Osmička	Překážka	Omezení rychlosti	Závod	Celkem
SlowFox MUNI Corp	200	150	150	400	900
KAW4Wheels	0	150	0	500	650
VAXNA	0	0	150	350	500
Richard's Gear	200	150	150	0	500
RoboticsBrno	125	150	150	0	425
Ligma	0	0	0	300	300
UNIZA_2	0	0	0	250	250
The Pegas	200	0	0	0	200
JAKOS	0	0	150	0	150
UNIZA_3	0	150	0	0	150
FENIX	0	150	0	0	150
RGM	0	0	0	0	0
SKALBO	0	0	0	0	0
UNIZA_4	0	0	0	0	0
UNIZA_1	0	0	0	0	0
SPIRIT	0	0	0	0	0
STARDUST	0	0	0	0	0
ZNTU_ITED	0	0	0	0	0
Infinite Synergy	0	0	0	0	0
PRIMA	0	0	0	0	0
Confuse-A-Robot	0	0	0	0	0
KNE_Januszomobilki	0	0	0	0	0
ZNTU_EPA	0	0	0	0	0
k4e	0	0	0	0	0
Overflown Boiz	0	0	0	0	0
ARMored Rider	0	0	0	0	0

Tabulka 7: Bodové hodnocení jednotlivých týmů v kvalifikačním kole NXP Cup



Obrázek 24: Odlesk na dráze

Název týmu	Osmička	Překážka	Omezení rychlosti	Závod	Celkem
ARCAR1	200	150	150	500	1000
ARCAR2	0	150	150	400	700
KAW4Wheels	150	150	150	250	700
AC TSM	125	150	150	100	525
The K-Team	40	0	150	300	490
Koala-Racer	100	150	0	150	400
Overtaker	75	150	150	0	375
VAXNA	75	150	150	0	375
CBP	0	0	0	350	350
MIDI-Unimi	20	0	150	50	220
EHTPCAR1	0	0	0	200	200
SlowFox MUNI Corp	30	150	0	0	180
HAL-9001	0	0	0	25	25
NASH	0	0	0	0	0
ENSIL-ENSCI	0	0	0	0	0
Richard's Gear	0	0	0	0	0
Tesla	0	0	0	0	0
GRKBrain	0	0	0	0	0
t-Car	0	0	0	0	0

Tabulka 8: Bodové hodnocení jednotlivých týmů ve finálovém kole NXP Cup

9 Závěr

Cílem práce bylo vytvořit software pro robotický model auta pro účast v soutěži Nxp Cup. V práci byly popsány jednoduché metody zpracování obrazu z kamery pro získání informací pro regulátor řízení. Dále byly popsány metody bezdrátového přenosu dat z modelu auta do počítače a jejich zobrazování v reálném čase. Zároveň byla popsána abstrakce programu a vysvětlena zkompilovatelnost systému pro různé platformy. Větší část práce byla věnována nejdůležitějším částem: zpracování obrazu, hledání čar v záznamu a řízení pomocí nalezených čar. Byl popsán systém pro detekci přejetí čáry pomocí trojice infračervených senzorů, který se ukázal jako jednoduchý a spolehlivý. I přes neúspěšnou implementaci původních myšlenek použití IMU jednotky se našlo smysluplné využití.

Vzniklý software hodnotím jako úspěšný i přes to, že robotické auto nezvládlo ani jednu disciplínu ve finálovém kole NXP Cup. Za úspěch považuji výsledek kvalifikačního kola, kde auto zvládlo všechny tři dodatečné disciplíny úspěšně.

Díky neúspěchu ve finálovém kole jsem získal různé postřehy, které můžou pomoci s dalším rozšířením této práce. Do budoucna by bylo vhodné obohatit auto o dvojici optických, nebo hallových snímačů pro detekci rychlosti otáčení kol. Zároveň by bylo vhodné rozebrat více do hloubky zpracování obrazu a najít způsob, jak vylepšit adaptivitu systému na špatné světelné podmínky. Dále by bylo vhodné obohatit model auta o systém radaru, nebo o dvojici ultrazvukových senzorů pro detekci překážek. Přestože je řízení pomocí dat z řádkové kamery při správném nastavení regulátoru dostatečně spolehlivé, mohly by se rapidně zlepšit výsledky použitím koncepce dvou kamer, kdy jedna kamera snímá bezprostřední okolí předku auta a druhá kamera snímá obraz ve větší dálce.

Literatura

[1] A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications. *Starlino electronics* [online]. [cit. 2019-04-26]. Dostupné z: http://www.starlino.com/imu_guide.html

[2] HEMANT, Kumar. Beginner's Guide to IMU. *Robotics Club IITK* [online]. [cit. 2019-04-26]. Dostupné z: http://students.iitk.ac.in/roboclub/2017/12/21/Beginners-Guide-to-IMU.html

[3] SOJKA, Eduard. *Digitální zpracování a analýza obrazů*. Ostrava: VŠB - Technická univerzita Ostrava, 2000. ISBN 80-7078-746-5.

[4] FRDM-KL25Z|Mbed. *Arm Mbed* [online]. [cit. 2019-04-28]. Dostupné z: https://os.mbed.com/platforms/KL25Z/

[5] FRDM-K66F|Mbed. *Arm Mbed* [online]. [cit. 2019-04-28]. Dostupné z: https://os.mbed.com/platforms/FRDM-K66F/

[6] *Freedom FRDM-K66F Development Platform User-s Guide*. 0, 02/2016. 2016.

[7] BOVIK, Alan C., ed. *Handbook of image and video processing*. San Diego: Academic Press, c2000. Academic Press series in communications, networking and multimedia. ISBN 0-12-119790-5.

[8] ZIEGLER, J. G. a N. B. NICHOLS. Optimum Settings for Automatic Controllers. *Journal of Dynamic Systems, Measurement, and Control*. 1993, **115**(2B), 759-768. DOI: 10.1115/1.2899060. ISSN 00220434. Dostupné také z: http://DynamicSystems.asmedigitalcollection.asme.org/article.aspx?articleid=1406231

[9] BLÁHA, Petr a Petr VAVŘÍN. *Řízení a regulace 1*. VUT Brno, 2005. Skriptum. Vysoké Učení Technické Brno.

[10] *The NXP Cup Official Rules Season 2018/19* [online]. In: . November 2018, s. 2 [cit. 2019-04-07]. Dostupné z: https://community.nxp.com/docs/DOC-335083

[11] *Using Accelerometers to Estimate Position and Velocity* [online]. [cit. 2019-05-01]. Dostupné z: http://www.chrobotics.com/library/accel-position-velocity

[12] What are good strategies for tuning PID loops?. *Robotics Stack Exchange* [online]. [cit. 2019-05-01]. Dostupné z: https://robotics.stackexchange.com/a/174

[13] HEWAGE, I.A. Premaratne a D.C. Hewage. Wheel Speed Control Algorithm for Rear Wheel Motor Driven Vehicle. *Journal of Engineering and Technology of the Open University of Sri Lanka*. 2016, **2016**(04), 12-26. Dostupné také z:

https://www.researchgate.net/publication/316133453_Wheel_Speed_Control_Algorithm_for_Rear_Wheel_Motor_Driven_Vehicle

[14] Ziegler-Nichols Tuning Rules for PID. *Microstar Laboratories, Inc.* [online]. [cit. 2019-05-01]. Dostupné z: http://www.mstarlabs.com/control/znrule.html

[15] HALOUNOVÁ, Lena. *Zpracování obrazových dat.* V Praze: České vysoké učení technické, 2009. ISBN 978-80-01-04253-3.